The impact of Software Product Lines from a Product Management Perspective

an Action Research approach into Software Product Line development

Yuri Sprockel
3363694
Master thesis of Business Informatics
Institute of Information and Computing Sciences
Utrecht University

Supervisors

Universiteit Utrecht

Dr. Inge van de Weerd
Email: i.vandeweerd@vu.nl

Prof. Dr. Sjaak Brinkkemper
Department of Information and Computing Sciences
Buys Ballot Laboratory, office 582
Princetonplein 5
3584 CC, Utrecht
The Netherlands
Email: s.brinkkemper@uu.nl

Michail Warrink
Product manager Marketing & Sales
CCV Holland B.V.
Westvoorstedijk 55
6827 AT, Arnhem
The Netherlands

Abstract

In recent years, software development has grown substantially and its products can be noticed and used on a global scale. Most organization nowadays, despite in which sector they operate in, use software products in order to support or excel their business. In addition, some organizations are founded based on software products (e.g. social media platforms) and business is not possible without. The main practice that manages the processes of the product software development is Software Product Development (SPM). It has been proven that in order for the software product to be successful in the given market(s), SPM should be properly in place. Larger organizations with global markets with software intensive systems apply a different development approach in order to be able to serve the market properly and be able to perform this in an organized matter. Such approach is called Software Product Lines (SPL) development and originates from non-software industries such as automotive. SPL development requires specific structuring of its process in order to benefit from its advantages.

In this research, we follow a well validated and already used at numerous Product Software Companies (PSC) Competence model, the SPM Competence Model and Maturity Matrix. This Competence model serves as the foundation of SPM for this research. Through an extensive literature study, we identified the key practices of SPL. These SPL-practices describe product management activities that are performed in a SPL development environment. In this research we evaluated the identified SPL-practices through experts active in SPL development and structured them accordingly through implementation into the SPM maturity matrix, making it the SPLM (Software Product Line Management) maturity matrix.

Afterwards, we applied the SPLM maturity matrix at a case company that develops SPL's. We noticed that PSC that develop SPL has specific capabilities' differences compared to PSC that do not develop SPL. This can also be noticed in the application of the SPM and SPLM maturity matrix. In addition to the capabilities in the matrix, we identified other aspects that are of great importance for the overall success of a SPL. We defined these aspects as 'Organizational Supportive Input' and these mostly focus on organizational aspects that should be performed to benefit the SPL processes. The key findings show that a PSC has to make a well-thought decision when deciding to engage in SPL development; per business functions such as Requirements management or Product planning specific SPL knowledge is necessary. The architecture of a SPL is vital for its long term success (an essence of SPL). PSC's which do not have the SPL architecture well defined upfront, will struggle later on. Finally, we propose the SPLM maturity matrix as specific version of the SPM maturity matrix for PSC with SPL.

Acknowledgements

This thesis is the result of the Master Degree of Business Informatics at the Utrecht University in The Netherlands. Before the start of my thesis, my interest in SPM grew that it led me to do Capital Selecta regarding Quality Requirements with a PhD. student from Lund University (Sweden). After a successful completion and a satisfied case company, I was certain that I wanted to pursuit a graduation assignment in SPM.

During my research I have been challenged by life and I haven't always to succeed the challenges the way I thought to. The loss of family members makes things harsh to continue with your daily routines. However, without the support of both the case company and Utrecht University I would even had the chance to finish my graduation. Therefore I'd like to show my gratitude:
To the case company, especially Michail my supervisor, I like to thank him for the opportunity for me to perform my research, his support, countless discussions and meetings and for always being ready to support my research and involve the necessary colleagues if needed. To Paul, Eric, René, Alfred, Bert-Jan, Jeroen, Arianne and Guus thank you for being wanting to be involved in my research, your collaboration and providing me with a proper working place in their office.
To Inge, thank you for your patience, advice and various improvement points during my research. Your insights have helped me understand the academic world much better. To Sjaak, thank you for your supervision. To Richard, thank you for your collaboration, it really helped my research.

During this research I can say I learned a lot, both on SPM and academic research as well as how organizations that develop software works. I certainly will be pursuing a career in SPM and will work unto the Product Manager function; it just drives me to oversee the processes of thinking of a product, developing it, bring it to the market and finally phase it out of the market, onto the next one.

Last but not least, I would like to thank my family and friends who supported me in unthinkable ways and believed I could finish.

Yuri Sprockel
April, 2013

Table of Contents

1 Introduction

In the last decade, products from most industries (e.g. telecom, software, automotive) have grown in functionality, complexity and responsibility (Bosch, 2002; Buhrdorf et al., 2004; Linden, 2002; Maßen, 2004; Northrop & Clements, 1999). This has excellent results for the end-users. It is the manufacturers who face the great challenges that come along with such development, such as product management. Software companies face similar situation. The process of developing software has evolved noticeably in the last 10 years (Vlaanderen et al., 2009a). Nowadays, software companies that manufacture large systems for large industries often make use of a so-called 'product line approach' like can be observe in other industries such as automotive. Product companies using a product line approach find that high quantitative improvements can be achieved in for instance productivity, time-to-market, product quality, and customer satisfaction (Northrop & Clements, 1999). With these great improvements comes great responsibility for the practice of software product management.

Software product management (SPM) has proven to be of utmost importance for products software companies. However, it is a challenging research area (Ebert, 2007; Weerd, 2009). The role of the software product manager has got little support from existing education and literature, yet the product manager is considered as the "mini-CEO" of the product software company (Bekkers et al., 2010). The product manager is responsible for a centralized position within the organization that communicates with all stakeholders and assures that all work towards the same goals according to defined strategies. Ebert (2007) has also pointed out that the success of software product is coherent with good product management. This requires a combination of business, technological and managerial skills that product management should possess. Fortunately, more and more researchers are becoming active in the SPM domain, based on the number of publications in recent years compared to before.

In order to have a solid foundation on the knowledge and practice of SPM, The Software Product Management Competence Model by Bekkers et al. (2010) is used. The SPM Competence Model represents all the areas that have been researched and identified as important in the SPM domain (Bekker et al, 2010). Explicitly, how SPM is understood and dealt with in this research is defined by the SPM Competence model. The SPM Competence model provides product software companies (PSC) and product managers a complete overview and information on related processes and stakeholders that are of importance. How these areas, processes and stakeholders relate to each other is modeled in the SPM Competence model, see Figure 1. Four main business functions are defined which described the main SPM functions, which are: Portfolio management, Product planning, Release planning and Requirements management. Each business function has a number of focus areas (white rectangles) that describes the strongly-related practices (referred to as capabilities) of that business function. The stakeholders consist of two categories, namely external stakeholders (e.g. customers) and internal stakeholder (e.g. Marketing or Development department). Since most PSC have, to a certain degree, software development activities one can expect that such activities might be modeled in the SPM Competence model. However, the activities of the development department are considered as a stakeholder that provides input to the business functions.

Product lifecycle management is one of the focus areas of Portfolio management implying, amongst others, information gathering and key decision making concerning product life and product modification across the complete product portfolio (Bekkers et al., 2010a). Product lifecycle also concerns the origin of products that do and do not result from software product lines (SPL). This is business function (Portfolio management) is one of the least researched of the model (Weerd, 2009). This lack of research triggered the original drive for this research.

The SPM Competence model corresponds with the SPM Maturity Matrix (Bekkers et al., 2010b) that is used to determine the current maturity levels of SPM processes of a PSC and provides an incremental improvement procedure. The SPM Maturity matrix is a matrix representation of the business functions and focus areas. In this master thesis, we used the SPM Maturity matrix to determine the current state of SPM processes at a case company with software product lines. What we are curious to investigate is the applicability of the matrix for organizations running software product lines, specifically.

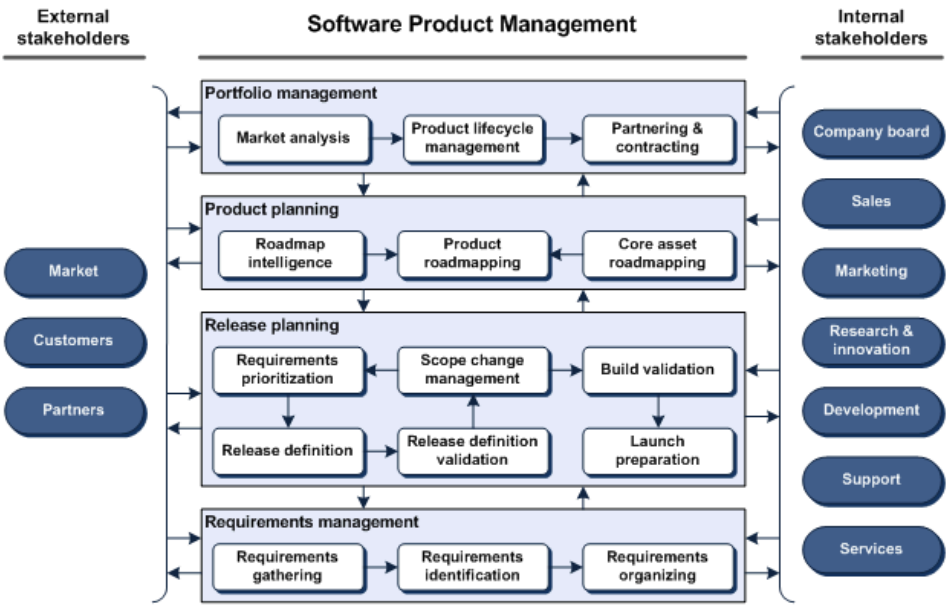

Figure 1: The Software Product Management Competence Model (Bekkers et al., 2010a)

1.1 Research trigger

As mentioned previously, four business functions are modeled in the SPM Competence model: Portfolio management, Product planning, Release planning and Requirements management. Out of these business functions, Portfolio management is the business function least (extensively) researched *(*Weerd*,* 2009*)*. During modeling of the SPM Competence model, the lower business functions, especially Requirements management and Release planning, received more research attention since these provide the fundamentals for the upper business functions. Recently, more research is focused on the other parts of the SPM Competence model; as this research does. However, no research on the model has yet focused on the practices of SPM when it concerns product lines. Corresponding with the upper part of the model, the case company (chapter 7) had most interest in Portfolio management (Product lifecycle management) as they were experiencing difficulties, primarily, with new product development. The case company describe this as *'The trajectory of a new product entering the concept phase (beginning of the product life cycle) to the point where requirements has to be gathered in order to build a release of the product to declaring an end-of-life for the product, needs structuring'*. At the moment this is performed according to their experience and knowledge which is not sufficient to excel at Software Product Management. For instance, requirements are not managed properly which leads to requirements being left out of major releases and product roadmapping was never a well-defined process which does not give a clear aim or planning on future product development.

From a scientific perspective, existing literature on SPL does not elaborately investigate the impact on SPM processes (see chapter 3) or SPM processes specifically for product lines. This triggered this research to investigate how the knowledge and practice of the SPM Competence Model compatible

is for product lines, i.e. the impact of SPL on SPM processes. In addition, most research on the SPM Competence model involved organizations with standard software products as well as some with SPL's. Evaluation of the model did not focus mostly on the difference of PSC's with product lines or PSC's without product lines. This research aims to evaluate the model for a PSC with software product lines.

From a practical perspective, the case company has little knowledge and experience in SPM practices. The current SPM practices grew from departmental best practices and some product managers have followed a professional course on Software Product Management at the Department of Information and Computer Science of Utrecht University. Currently, the case company struggles with software development when it comes to the product management processes, e.g. requirements identification for new products, requirements selection for next release, product roadmapping and product life cycle current products. There is enough room for improvement with respect to structuring and improving SPM processes for the case company.

1.2 Problem statement

Software product lines are getting popular and attractive for organizations with large software-intensive systems and the benefits are clear: less development and maintenance costs, faster time-to-market, improved product quality, improved customer satisfaction, reuse of artifacts such as architecture and more (Northrop, 2002; Bosch, 2002; van der Linden, 2002; Birk et al., 2003). However, these advantages claim a remarkable amount of effort from a SPM perspective that is poorly defined. Bekkers et al. (2010) have constructed a competence model (SPM Competence model) that describes the most important areas in SPM which numerous PSC find substantially helpful and provides structure in their SPM practices.
Existing literature on SPL suggest little relation to the SPM Competence model or any SPM body of knowledge for that matter, not forming an association between SPM processes and SPL-practices, whilst software as a product is still the end-result. Accordingly, the main research question of this master thesis is:

"How can product software companies structure their software product management processes according to their product line development approach?"

1.3 Research contribution

The main objectives of this research can be directed towards a scientific and a practical objective, respectively:

Scientific contribution
The main scientific contribution of this research is the evaluation of the applicability of the SPM Competence Model for product lines. This implies to what extent the SPM Competence model would be applicable for PSC with product lines. This research will identify which part of the model, i.e. business function, focus area, capability and stakeholder, applies for SPL and which part less or not. In addition, this research will also identify SPL-practices with respect to SPM that are not included in the SPM Competence model or the maturity matrix. These SPL-practices will be translated to fit in the maturity matrix and evaluated during this master research. Thus, the SPM Maturity Matrix will be enriched with specific SPL knowledge, namely specific SPL capabilities. This SPM maturity matrix for SPL can be used at other PSC with SPL. Furthermore, the relation of SPM with SPL is not extensively elaborated in scientific literature which makes it rather exciting combination of topics. After this research, a step is put towards SPM for SPL as the product management processes gets evaluated for SPL. Finally, this master thesis is another validation of the SPM Competence model and the maturity matrix (Bekkers et al., 2010a).

Practical contribution
The main practical contribution is the knowledge and advice on how to structure SPM processes for SPL developments for the case company. Under SPM processes we understand the SPM business functions and focus areas as defined by the SPM Competence model. This knowledge and advice is given in the form of textual and graphical recommendations with the aim to maintain full control and execute the processes according to a well-organized structure. As a result of performing the SPM maturity matrix, the recommendation can be tailored according to the desires of the case company through incremental improvements.

1.4 Terminology

During this master thesis research, the domain of Software Product Management is central and for this reason some of the key terms are further explained, beforehand, in this section. The descriptions and definitions used are most applicable for this research.
Ebert (2007) defined product management as "the discipline and role, which governs a product (or solution or service) from its inception to the market/customer delivery in order to generate the biggest possible value to the business". In a later research Ebert (2009) adds that product management covers all product life cycle phases.

SPM concerns the process of managing product software in a broad sense, as described above. Xu and Brinkkemper (2005) defined product software as "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market."

Clements and Northrop (1999) define a software product line as "a set of software-intensive systems that share a common, managed feature set satisfying a particular market segment's specific needs or mission and that are developed from a common set of core assets in a prescribed way". Both product software and SPL has a market focus. However, SPL has a strong focus on using core assets to develop product software. Another deviation is that SPL describes also the 'way of developing', i.e. for instance ".. a common set of core assets in a prescribed way".

As mentioned before, the SPM Competence model describes business functions, focus areas which represent a grouped together set of relating capabilities. These are all presented in the SPM Maturity Matrix. During this master thesis, often 'SPM Competence model' and 'SPM Maturity Matrix' are used rather interchangeably, since the SPM Competence model is often the presentation to the world and the SPM Maturity Matrix more the technical view backing the model up. The SPM Competence model and SPM Maturity Matrix will be explained in detailed in the literature section.

1.5 Thesis structure

Below, we explained briefly how the structure of this thesis is organized.

Chapter 1 introduces the reader into the domain of SPM and SPL development in order to get acquainted with the context of this research and the terminology. In chapter 2 we present the research questions and discuss the methodology applied during this research. We also discuss the validity issues. The literature study is presented in chapter 3. In chapter 4 we will process the literature findings into useful concepts such as SPL-practices and SPL-activities. The expert evaluation of the useful concepts is presented in chapter 5. In chapter 6 we process the feedback of the evaluation and chapter 7 we present the case study. Chapter 8 we present the conclusion of this research, the shortcoming and directions for future research.

2 Research approach

In this section we elaborate on the research triggers, the sub-research questions supporting the main research question, the applied research method, and the validation of this research.

2.1 Research questions

Here, we briefly restate the main research question and introduce the sub-research questions:

"How can product software companies structure their software product management processes according to their product line development approach?"

Supporting the main research question is five sub-research questions presented below. By answering these questions enough knowledge will be gained in order to answer the main research question. Below, the sub-research questions are described and elaborated upon.

2.1.1 SRQ1

Which key practices can be identified in software product line developments, from a Software Product Management perspective?

The basics of the subject (SPL) must be known in order to understand it. This means identifying the key practices of SPL development. These practices serve to give proper knowledge on what SPL actually is and how organizations practice it. Questions such as *What is SPL? What is known on SPL development? How is it performed? How does science describe this phenomenon? What is the difference with standard product development (non-product line)?* will be answered. However, the focus lies on product management which means that the actual development process of the software, the writing of the software code and the necessaries, is not included in this researched.

2.1.2 SRQ2

What different Software Product Line development situations can be distinguished that influence product management processes?

The aim here is to identify the different situations (or contexts) in which SPL developments are initiated. This can also be interpreted as the approach used to organize the managing processes concerning development, such as different project types with different purposes (e.g. bug fixes projects or new features implementation projects). The focus is on the influence that each of the different development situations has on the product management processes (business functions, focus areas, capabilities and situational factors).

The main challenge is to research *what the circumstances are that triggers these SPL developments. When does a PSC decide to start developing and Why (what is the motive)? What drives this development? Is it business cases? Is it regular maintenance or bug fixing?* Questions like these are essential in this sub-question.

The findings of SRQ2 should be mapped to the SPM domain with the aim to present the influence certain development situations (e.g. project types) have on the SPM/SPL processes. However, these are less straight-forward compared to the SRQ1 findings, where similar concepts are compared. The situation findings are expected to be information that represents different grouping of activities which cannot be compared to any other grouping, since they simply do not exist in the SPM

Competence model, i.e. cross business functions groups of capabilities. For instance, with respect to the Competence model this would imply that particular capabilities from different focus areas and business functions are grouped. On the other hand the identified SPL development situations can also be more elaborate than just a grouping of activities, which may lead to a more challenging mapping task.

2.1.3 SRQ3

How can the obtained knowledge be used to structure Software Product Management for product line developments?

The aim of this sub-research question is to present and apply the obtained knowledge on SPL, i.e. SRQ1 findings (and SRQ2 findings), effectively. This means that the SRQ1 findings need to be applied in such a manner that it can be consulted when structuring is needed, with respect to SPM for product lines. The purpose of the obtained knowledge is similar to the information of the SPM Competence Model; mainly to give knowledge on the important processes of product management for SPL, in a structured manner.

2.1.4 SRQ4

How applicable is the Software Product Management Competence Model for software product line development?

The aim is to find out to what extent the SPM Competence, and thus the Maturity Matrix, model applicable is for SPL, since the model is being used as the reference for how we understand and deal with SPM in this master thesis. Analysis of the SPL practices, i.e. SRQ1 findings, and comparison with the SPM Competence model points out the possible commonalities and differences. Depending on the commonalities and the differences and indication can be made on the degree of applicability of the model for SPL. The greater the amount of the commonalities, the higher the applicability of the model for SPL will be. In contrast, the greater the differences, the lower the applicability of the model for SPL will be.

2.1.5 SRQ5

How can Product Software Companies use the gained knowledge to improve their Software Product Line Development approach?

The aim is to be able use the gained knowledge in this research for other PSC in some way. This means that while processing information in order to answer the previous sub-research questions, other PSC has to be considered. The gained knowledge has to be made as common as possible and less PSC-specific.

2.2 Research method

The methodology followed during this research is depicted in figure 2 in a Process-Deliverable Diagram (PDD) (Weerd et al., 2009). PDD is used as a meta-modeling technique that is used by situational method engineering where a process view and a deliverable view are presented. Figure 2 presents the processes as well as the deliverables of the processes performed.

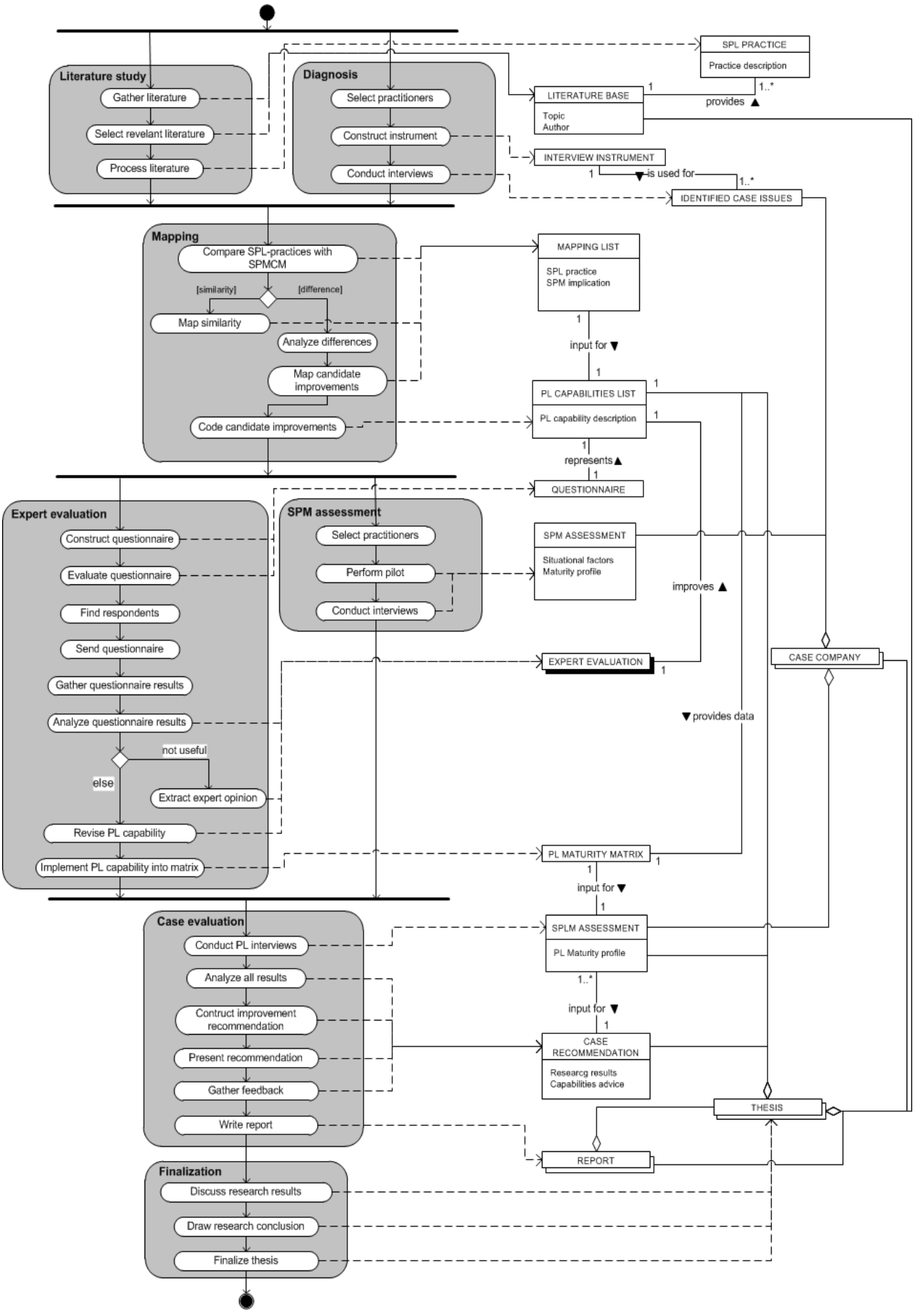

Figure 2: The research Process-Deliverable Diagram

In performing this research, we employ Action Research as a research method. Baskerville (1999, 2004) describes Action Research (AR) as a method that aims to “solve current practical problems while expanding scientific knowledge”. Henfridsson et al. (2007) adds to Baskerville description, stating that the rationale of AR is that the researchers should support the practitioners in real-life problem solving and whilst directly involved in the organizational change should be able to increase scientific knowledge. This implies that AR strives to, collaboratively, improve understanding of a phenomenon, improve professional practice, the practitioners and the situation it takes place in; and whilst doing this to learn from the improvement process.

Participatory action research is distinguished by the additional characteristic involvement of the practitioners as both subjects and co-researchers (Baskerville, 1999). "It is based on the Lewinian proposition that causal inferences about the behavior of human beings are more likely to be valid and enactable when the human beings in question participate in building and testing them" (Argyris and Schön, 1991). Action science is distinguished by the additional characteristic of a central emphasis on the spontaneous, tacit theories-in-use that participants bring to practice and research. Action research aims for an understanding of a complex human process rather than prescribing a universal social law.

The aim of this research is to engage in a real-world SPL development environment to be able to understand and improve the practice thereof, from a SPM perspective. Hence, we chose AR to be able to apply the SPM Maturity Matrix, and thus validate the SPM Competence model, in real-world SPL situations. In addition, AR is one of the few valid research methods that can legitimately be employed to study the process and effects of particular system developments approaches in human organizations (Baskerville and Wood-Harper, 1996). In combination with AR, a case study is an ideal approach when little theory is known on the subject (SPM for SPL) and the subject is broad and complex (Dul & Hak, 2008). According to Yin (2009), case studies are relevant in the case of explaining how certain phenomenon operates which is the case in this research.

When conducting AR, a sequence of steps has to be followed that is referred to as a cyclical process that is usually guided by a lead-researcher (Baskerville, 1999). The steps that constitute AR can be seen in figure 3 and these are:

1. Diagnosing
2. Action planning
3. Action taking
4. Evaluation
5. Specifying learning

Only the last step, Specify learning, is performed solely by the researcher. The other steps are performed in (partial) collaboration with practitioners. In the following sub-sections each step is elaborated upon.

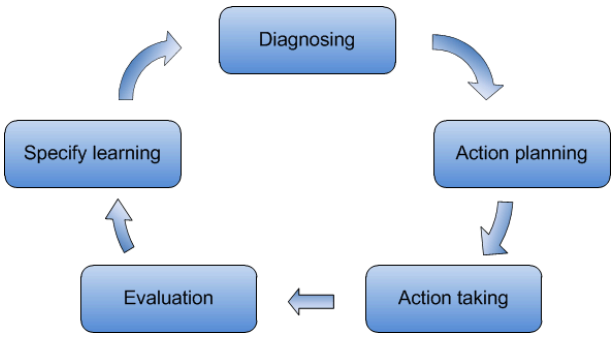

Figure 3: Action Research steps (Baskersville, 1999)

2.2.1 Diagnosing

The Diagnosing step implies the identification of the fundamental issues that drives the organization's desire for change. This step entails a self-understanding of the issues the organization is dealing with in a wholeness manner, i.e. not excluding possible related factors (Baskerville, 1999).

Parallel with the first steps of diagnosis, a literature study on the matters at stake (mostly SPM and SPL) was initiated. The purpose of the literature study is to give the basic and extended theoretical foundation necessary to engage in further steps of the research. For this research, the first steps of diagnosis were the attendance (by the researcher) of certain meetings. The first was a 'new-product-development-project' meeting to witness how the very first steps of a product development project are taken. This was practically a brainstorm session between upper and lower-management personnel, i.e. Business Development managers, Product managers and Technical managers. In addition, the researcher also attended the monthly international product management meetings for the same purposes, except it is for product development on international level implying a much wider scope. This was similar for the progress product development meetings. These meetings gave inside information of the product line development practice and its practitioners and the reasons and desires for possible changes, as well as details that can be related to the SPM Competence model, e.g. product roadmapping challenges, release planning not considering mandatory requirements and issues in requirements organization. Document analysis also formed part of the diagnosis step. Examples of documents are: presentations of new product developments (software and hardware), product roadmaps, software budget allocation, international projects development, software management processes, and more.

With all the information shared, some needed to be further explained by practitioners with specific knowledge and expertise. The product manager pointed out the necessary experts (3) of whom the researcher held interview sessions with, inspired by a semi-structured interview, i.e. a list (see Appendix E1) with pointers grouped by categories on SPM and SPL developments was created that was discussed during the interview. The interviews were divided into, one regarding software development from a parallel development perspective, and two regarding software development from a product development perspective.
The SPM Competence model was globally explained to all the practitioners. As mentioned previously, some practitioners and product managers followed a professional SPM course; they were familiar with the model and other could have, partially, related to it in their daily tasks. However, the SPM Competence model was seen as the ideal situation and that the practitioners and their practice would have to improve significantly in order to reach most of the competences proposed by the SPM Competence model.

From the diagnosis step the researcher was able to create the complete product portfolio, generic system architecture and the developing product lines at the organization. These added to the fundamental understanding needed. Table 1 gives an overview of the activities performed in the first step of AR and the deliverables they resulted in.

Table 1: Activities and Deliverables of Diagnosing step of Action Research (Baskerville, 1999)

Activity	Deliverable
New product development meetings	Complete Product Portfolio, Generic system architecture, Product Lines overview.
International product management meetings	
Monthly progress product development meetings	
Document analysis	
Expert interviews	Detailed information on the Custom Parallel Development and Issues definition

Table 2 presents the issues that were identified during the diagnosis step, regarding the organizational desire for change. Some of these issues are described under the business function (The SPM Competence Model) which they resemble most. The other types of issues that were also identified were more project management, organizational or technical –related. These are described under 'Diverse'.

Table 2: Identified issues defined.

Business function	Issue description
Requirements management	Customer requirements are left out from major releases. This point out that communication of the release definition is not validated properly, meaning a re-confirmation of the (prioritized) requirements that will be implemented in the next release.
Requirements management	Not all the stakeholders are connected to the requirements management system (JIRA), i.e. the stakeholders that are responsible for new functionalities for the products. Stakeholders using the system, development and maintenance departments, benefit from more requirements management possibilities, such as requirements lifecycle management. Hence, communication of requirements is troubled at times.
Requirements management	Some market requirements are not (properly) translated to product requirements. This makes it vague for development to understand what has to be implemented, the purpose of the requirement. In addition, these vague requirements are included in the planning and roadmap. By the time it gets to development, they are under-specified and not feasible.
Release planning	Defining the next release, i.e. prioritizing happens poorly and making a selection which requirements will be implemented in the next release becomes even more challenging (to reach an agreement). Due to limited engineering capacity and budget allocation, not all requirements from all stakeholders can be in the next release. In addition, usually development finds the release definition difficult to realize given the time they have to deliver.
Product planning	Current roadmaps of the case company give little certainty that the products will be delivered on the planned time. Currently, the roadmaps include products that are hardly feasible (time-to-market) and it should be more transparent (better estimations and specifications). In addition, if the product planning/roadmapping is more accurate (it's clear what features and what product has to be developed), then requirements management and release planning are easier performed. *"For instance, when you communicate the roadmap to Account managers, they have to know when products are coming out with what features in order to sell, which means the actual product have to be delivered or customers are not satisfied and that can bring issues and a bad image for the case company".*
Portfolio management	Partnering with the supplier can be better. The case company wants knowledge on future products that will be brought out by the supplier. This is beneficial, since the case company can plan development ahead. At the moment, the case company approach towards its supplier is quite passive. However, products that the supplier will no longer develop or support anymore are properly communicated.

Diverse	Issue description
Software architecture not structured efficiently	The software architecture is in such a way currently, that what should be relatively easy modifications require software code from multiple modules to be altered. This takes development substantial amount of extra time. In addition, the architecture also limits development possibilities.
Release definition cost too much time	The cause is that practitioners that are dealing with long term issues are burden with short-term issues that management finds have higher priority, at that time. In addition, the long term issues are postponed and it piles up making it harder to decide on certain issues. Mostly, this boils down to deciding which requirements to implement in strategic developments, i.e. major product releases with the intention to possibly penetrate a new market.
Merging existing products in main product line	The plan is to have fewer product lines with the different product variants under it. Merging the 'loose' products from each country (stakeholders) in main product lines is a great challenge, since from each stakeholder generic software needs to be extracted and implemented in the main tree without conflicting with other products from other stakeholders. Next, product-specific software code needs to be distinguished the products from one another.

2.2.2 Action planning

In the Action planning step, the collaborative work starts between the researcher and the practitioners in the real-world practice. This step entails specifying actions or activities, on organizational level, that would be candidate for solving or improving the fundamental issues (Baskerville, 1999). The specified actions are planned according to the desired future state of the organization, the target, and according to an approach that fits the organization best.

The collaboration started between the researcher and the product manager of the Marketing & Sales department. In order to fully comprehend the organization and its way of working, regarding SPM and SPL development, bi-weekly sessions were held where the researcher and the product manager discussed the theory (preliminary results of the literature study and document analysis), the current issues and the pragmatic solution applied by the organization.

As a result, it became clear that the organization employ a self-created development approach, so-called 'Parallel Development', which allows multiple developments to take place in parallel. This parallel development is something to take into consideration for the SPM Maturity Matrix, since experts mentioned that each development 'stream' has its own characteristics and limitations. Consequently, the literature study was extended to also include product development situations or contexts (project types or 'streams'). These product development situations would have influence on the structuring of the SPM processes.

The SPM Competence model together with the Maturity Matrix were discussed (real world examples) with the product manager as an assessment tool to be applied at the organization (the action). The Maturity Matrix would clarify the current levels of the software management processes, according to the SPM Competence model. Knowing the current state will act as a starting point for the desirable organizational change, with respect to structuring and improving the SPM processes. However, the focus on SPL cannot be excluded. Hence, a proposed course of action of a SPM Maturity Matrix dedicated on SPL (section 2.2) was confirmed as a plausible solution by the product

manager. This SPM Product Line (PL) Maturity Matrix would give expert knowledge on specific SPL product management practices.

The application of the SPM Maturity Matrix would take place in the form of interviews. In order to get a total and reliable response, multiple perspectives were needed. The researcher discussed this with the product manager and came to the solution that the expertise of certain practitioners was not to be missed. This implied that experts from different departments were involved to complete the interview with the purpose that all the all the expertise needed were included in the research. The included departments were:

Table 3: Selected practitioners for interview.

Practitioner	Department	Function
P1G	Marketing & Sales	Business development manager: responsible for creating business with (new) customers
P2M	Marketing & Sales	Product manager: responsible for creating business with dedicated focus on product management
P3A	Product development	Project manager: responsible for product development projects
P4G	Product development	Product development manager: responsible for specialized products development projects
P5A	Product development	Senior architect: responsible for most system architectures and vital components such as Security.
P6J	Maintenance	Product quality manager: responsible for product quality aspects and requirements
P7R	Maintenance	Product manager: responsible for products' maintenance and fixing in the field
P8B	Maintenance	Product quality manager: responsible for product quality aspects and requirements

Firstly, the literature results on SPL-practices would need to be inspected to identify practices that are related to SPM. Secondly, the selection of the identified practices that relate to SPM need to be mapped to the SPM Competence Model, i.e. business function, focus area and capability. These practices can then be either a similarity or a difference, in which they are then candidates for improvement for in the SPM Maturity Matrix. Thirdly, evaluation of the matrix is needed before applying it at the organization. Once evaluated and the necessary changes has been made, the SPM Maturity Matrix with SPL-practices can be applied at the organization and the assessment includes both the original SPM and the SPL specific practices. The results of both assessments can be compared during analysis and used for recommendation.

Table 3 gives an overview of the activities performed in the Action Planning step of AR in this research. The meetings introduced in the Diagnosing step, continued in the Action Planning step as well. However, completely new deliverables were not necessary. Table 3 also gives an overview on the decided planned activities that have purpose to achieve the desired organizational state.

Table 4: (Planned) Activities and Deliverables of Action Planning step of Action Research (Baskerville, 1999)

Activity	Deliverable
Meetings from Diagnosing step	-
Literature study	SPL-practices and SPL development situations.
Bi-weekly discussion session	Feedback, practical information, and discussion and decision-making on plausible actions for change.

Planned activity	Deliverable
SPL-practices identification of SPM relation	Commonalities and differences with the SPM Competence Model
SPL-practices mapping	SPM Product Line Maturity Matrix
Application of SPM Maturity Matrix	Current SPM Maturity profile
Evaluation of SPM PL capabilities	Expert evaluated SPM PL capabilities
Application of SPM PL Maturity Matrix	SPM PL Maturity profile

2.2.3 Action taking

In this step, the actions planned previously are implemented. The researcher and the involved practitioners collaborate as needed for this step.

The actions planned are put into action:

- SPL-practices identification of SPM relation
  The literature results on SPL-practices are investigated to find possible relation to SPM, since this is the main focus of the research. The essential actions of each SPL-practice are registered in a similar way that the SPM Capabilities are so these two can be compared easier. Software product lines are a broad topic, as can be seen in literature of previous researches. However, in this research we are interested in the product management related practices of software product lines. Therefore, we are interested in the SPL-practices that can be related to the SPM. Once the relation is known, i.e. there is a relation so it can be mapped or no relation is identified (e.g. too development related) and it cannot be used in this research. See chapter 4 for further explanation.
- SPL-practices mapping
  The identified SPL-practices are mapped as similarities, candidate improvements or neutral. Similarities are SPL-practices that are similar to the concepts (business function, focus area or capability) described in the SPM Maturity Matrix. Candidate improvements are SPL-practices that are product management related with specific link to SPL. Neutral SPL-practices are nor a convincing similarity nor a convincing candidate improvement. The candidate improvements get the most interest, since they bring specific product line knowledge into the maturity matrix. See chapter 4 for further explanation.
- Application of SPM Maturity Matrix
  The SPM Maturity Matrix assesses the current SPM maturity at the case company. Practitioners from 3 different departments are interviewed in order to cover input from all expertise areas. The result (SPM maturity profile) shows where the case company is excelling in its SPM processes and where it is performing less. The SPM maturity profile will be compared with the SPM maturity profile from the SPLM Maturity Matrix. See chapter 6 for further explanation.
- Evaluation of SPLM Maturity Matrix
  The mapped SPL-practices have to be evaluated before being implemented in the maturity matrix. For the evaluation, only the candidate improvements were used, since these are possible additions to the matrix. The candidate improvements are presented as SPM product line capabilities in a questionnaire. The questionnaire will be sent to product line experts abroad who gave their opinion on the usefulness of PL-capabilities. Based on the evaluation feedback, PL-capabilities were, rewritten, implemented in the SPM Maturity Matrix or discarded. See chapter 4 for further explanation.
- Application of SPLM Maturity Matrix

Similar to the way the original SPM Maturity Matrix, the SPLM Maturity Matrix is performed. The same practitioners from the same departments were interviewed again this time with the difference of the presence of the PL-capabilities. See chapter 7 for further explanation.

2.2.4 Evaluation

The evaluation step enquires whether the implemented actions were the lone cause for solving the organizational issues and not regular or routine actions of the organization, i.e. where the implement actions were successful. Where the implemented action was not successful, notations have to be made on the corresponding action and issues for the next AR cycle.

The researcher and the practitioners evaluate the results of the actions taken, collaboratively (the implemented SPL-capabilities). This entails confirming whether the SPL-capabilities would have impact on the desired organizational changes in a positive manner. Each SPL-capability is discussed one-on-one between researcher and practitioner. Next to the SPL-capability being implemented or not, it is discussed whether it would be useful for the case company and in which context. This follows the expert evaluation of the SPL-capabilities before implementation into the maturity matrix. Since actual organizational changes cannot be implemented by the researcher, this is the best possible form of evaluation obtainable on the to-be implemented changes. Recommendations based on the results of the Maturity matrices will presented to the case company.

2.2.5 Specifying learning

While the five steps of AR have a cyclical nature. The step 'Specifying learning' is an ongoing process (Baskerville, 1999). The researcher takes the evaluation of the previous step and analyzes the data and content gained.

The analysis involves identifying and marking interesting sections in the data, mainly on the SPM Maturity matrices and feedback on the SPL-capabilities. Explicit or concealed sections would be candidate parts to be declared as lessons learnt, focusing on a more general organizational and a scientific perspective. Whether the actions were successful or not, gained knowledge can be used for structuring organizational standards, with respect to SPM and SPL. Where the actions were unsuccessful, the additional knowledge that is gained can be used as input for the Diagnosing step and further Action planning with more specific focus and aim to be more successful. Unfortunately, a second AR cycle is not feasible during this research. See chapter 8 for further elaboration.

2.3 Validity

Conducting this research, we considered the four validity threats as described by Wohlin et al. (2012). The validity threats are applied to the research in general and more specifically on the case study at an organization. Dul & Hak (2007) defined a case study as *'a study in which (a) one case (single case study) or a small number of cases (comparative case study) in their real life context are selected, and (b) scores obtained from these cases are analyzed in a qualitative manner'*. A case study is used especially when a phenomenon, in this case SPM for product lines, still needs investigation on the boundaries of the objects of the study and the context of the study. Hence, Action Research is combined with a case study. Below, we elaborate more on the validity threats.

2.3.1 Conclusion validity

The conclusion validity is concerned with the ability to draw accurate conclusions. A part of conclusion validity regards statistical aspects which are not relevant for this research. Regarding the

interviews sessions, these were conducted in one uninterrupted session between researcher and practitioner. The occasion was informal in order to prevent any kind of pressure on the practitioner, however the researcher prevented the focus of the interview to deviate to other less related topics. In addition, no discussion with a third party was possible that could have influence the interview. However, it is possible that in between the two interview sessions of the SPM Maturity Matrix and the interview sessions of the PL Maturity Matrix practitioners discussed their opinion with one another. This is challenging to prevent.

Measurement validity was covered by the interview instrument, i.e. the maturity matrices. Some of the posed questions were semi-closed (three optional answers possible), however, the majority was closed questions with a possibility to give a rationale on the question asked. The instrument ensures that meaningful data is collected regarding the ideas contained by the practitioners regarding the corresponding concepts, e.g. focus area or SPM capability.

To ensure understanding and quality of the interview setting and instrument, a pilot was performed with the product manager responsible for the selection of the subjects. The interview started with an introduction on what the aim was an example questions were showed where after the actual questions started. This procedure was completed in the same way with every practitioner. The location was also kept the same, i.e. the same meeting room was used for all interviews which alleviates the issue of irrelevancies.

2.3.2 Construct validity

Construct validity is concerned with the relation between theories behind the research and the observations made during the research.

By using eight different subjects representing 6 different roles, the issue of mono-operation bias was alleviated. Mono-operation bias is when the research includes only a single independent variable, case, subject or treatment (Wohlin et al., 2012). This causes the main construct of the research to be under-represented. However, the issue of mono-method bias remains. The main method used for measuring was interviews. After the pilot, it was clear that even though questions were (semi) closed, practitioners will have the need to explain their answer. This possibility was implemented in the instrument. For every answer, the rationale can be used for cross-checking the answers with other practitioners.

The introduction of the topic ‘Product Line Architecture’ brought a threat with it. One respondent mentioned in the questionnaire that he had no experience with the topic and therefore could not answer the corresponding questions In the second interviews session only 2 (of the 8) practitioners had good knowledge about and experience with. This implies that Product Line Architecture is the topic with least evaluation compared to the other capabilities.

The guarantee of complete anonymity alleviates the issue of evaluation apprehension, i.e. their answers was only going to be accessed by the researcher and not showed or discussed with other practitioners. It was made clear to the practitioners that their purpose was to give input purely on their knowledge and experience and to simply mention if they do not know or are not sure about their answers.

2.3.3 Internal validity

The internal validity threats are related to issues that may affect the causal relationship between treatment and outcome. The time difference between interviews of the SPM Maturity Matrix and the SPM PL Maturity Matrix is nine - ten months. A difference in awareness was noticed, namely the interviewees were more aware of their own responsibility and daily tasks regarding product management. Within departments some improvements in process structuring was taking place. This

rather small change would have not been noticed if the time between the interviews of the maturity matrices were less, e.g. two – three months.

The interviews were all conducted using the same instrument, i.e. the SPM Maturity Matrix and the SPM PL Maturity Matrix. The PL version is an extension from the original SPM Maturity Matrix, which has been used in various organizations (Bekkers & Weerd, 2010), which alleviates the threat of instrumentation. Both are represented in a Microsoft Excel sheet. The difference is that the PL maturity matrix has the PL-capabilities. The results of the first interview sessions (SPM Maturity Matrix) were not discussed with the interviewees in order to prevent unintended learning. The first interviews session took place in a random order, i.e. as the practitioners were available. For the second interviews session, we planned to have the same order as the first session. Unfortunately, due to time limitations and busy schedules of the practitioners this could not happen. In return, the practitioners were interviewed as they were available. The main reason for this was to prevent practitioners to inform one another (unintended learning) and have the exact same circumstances as the first interviews session. However, one subject (interviewee) showed some signs of maturation threats during the first interviews session; the subject was getting tired. This was covered with a bathroom and coffee break. Moreover, the second interviews session were kept shorter by focusing on the PL-capabilities, all under 60 minutes, in order to prevent maturation.

2.3.4 External validity

External validity concerns the ability to generalize the results of the research (Wohlin et al., 2012). In this case, this is the applicability of the PL Maturity Matrix in organizations other than the case company. The essence of qualitative research, i.e. more concerned with understanding and explaining the phenomena at stake, makes it impossible for complete replication since identical circumstances can hardly be recreated nor there is no population to generalize to. In order to have more generalizable results, multiple case studies should have been performed in similar context and circumstances as at the case company. However, some of the triggers of the research are recognized at other organizations. This is noticed through the PL-capabilities questionnaire that was received from the expert organizations responsible for the evaluation of the PL-capabilities. Some responses are very similar to that of the case company. Since the questionnaire is filled in by three experts from two distinct organizations, it is not enough to adequately apply the results of this research to other organizations. However, the PL-capabilities are firstly described from multiple existing literatures covering various researches at various organizations in order to keep the description as general as possible. After evaluation, some PL-capabilities are rewritten accordingly. In addition, between the case company and the expert organizations little to some differences can be noted.

As for the interviews for the SPM (PL) Maturity Matrices, different practitioners, covering the business functions and focus areas, from different departments are selected in order to receive the proper input based on their expertise and knowledge. The interviews' setting is identical for each interviewee.

External validity has a relation with reliability. The reliability of a research refers to the ability of reproducing the results. Taking the participation aspects of AR into consideration (situational and context bound), it can be stated that given any organizational situation at a particular time, with its particular participants having their own individual or shared opinions, may be unique, it cannot be guaranteed that results can be made richly meaningful to people in other situations (Checkland et al., 1998). However, by properly describing the recoverability of the results (epistemology), the content of the research can be more recovered and it will justify the extent of generalization of the results of AR. This is achieved by stating '*the epistemology (the set of ideas and the process in which they are used methodologically) by means of which they will make sense of their research and define in that matter counts for them as acquired knowledge'* (Checkland et al., 1998). This is covered by the steps of AR and the documenting thereof. Each step is described how it is performed with its deliverables

in this research. Mainly through the last step of AR, Specifying learning, the results regarding knowledge gaining is covered. The evaluation on the successful and not successful implemented actions is used as source for acquiring knowledge. Whilst the successful implemented actions will account for acquired knowledge on an organizational level, i.e. the knowledge can be meaningful for other organizations, the unsuccessful implemented actions will account for acquired knowledge that will be needed as input for the Diagnosis and Action planning steps of the next AR cycle, i.e. future research.

3 Related literature

In this section, we elaborate in detail on the results of the literature study mentioned previously as well as literature relevant to this research. The content of this section is presented in such a way that the more global subjects are explained first and the more dedicated subjects are explained thereafter. First, we elaborate on how the literature was gathered and selected before we used it for the research.

Literature gathering
Literature on SPM forms the foundation for this research as this research takes SPM as described by the SPM Competence Model. The other part of literature is regarding SPL. Literature on SPL was gathered through online academic search engines, mainly Google Scholar (also Omega and Citeseer). The main keywords used for the search were: Software product lines, Software product lines management, Software product family, Software product family management, Product management Software product lines, Software product lines life cycle, Product line management, Product family management, Software product management for product lines, Product management and product lines, and other combinations. Another topic that was included into this research was on Software product line development situations. The case company had interest to know how software development situations can influence SPM processes, e.g. Time-to-Market development projects or Maintenance development projects.

A great amount of the keywords yielded the same results. Literature was gathered based on: title, citations, short description and where it was published. The titles of the most related literature were equal to the keywords. However, the less related literature had vague titles and not similar to the keywords searched for. Here, the short description was useful, citing three lines of the literature. Besides a relevant title, citations show how popular the literature is. Relevant literature was gathered that had 15 citations or more. Journal, conference and workshop literature had preference in that order to be gathered. White-papers or literature that was not published were not collected. Based on these gathering criteria's a total of 71 papers were gathered.

Literature selection
Once literature was gathered it was time to process it and discard literature that was irrelevant.
Literature was mostly selected based on the Abstract, Introduction and Conclusion of the paper. These three sections give a sufficient enough overview to assess whether the literature is proper for the research or not. Papers that were not clear after those three sections were read in more detailed as needed. A total of 56 papers were selected (15 discarded). Literature that was deviating too much to embedded systems and design as well as non-information system related products were not selected. Exceptions were made for SPL Development situations, since this was very difficult to find purely based on software. Most found literature regarding this topic was too focused on Project Management and business-related. Literature that was published prior to 1990 was also no selected for the timespan and evolution of information systems from then to present.

Below, we firstly present literature on SPM as defined by the SPM Competence Model, since this constitutes the foundation of this research. Secondly, we present literature on SPL. These are the literature most relevant for this research, with respect to SPM, and common in the literature base as multiple authors have researched similar topics. Thirdly, we present literature on software development situations and organizational structures we found most useful.

3.1 Software Product Management

Software product management (SPM) is the main knowledge domain within this research. Weerd et al. (2006) state that SPM is getting more and more dedication in product software companies,

especially with the competitive software markets of nowadays that demand skilled and proficient product management for a product to be successful, e.g. Microsoft (Cunsumano et al., 1995). However, traditional product management has been of strong strategic importance for decades, especially in the manufacturing sector (Kilpi, 1997). SPM has its advantages in contrary to traditional product management, e.g. software can easily be copied and sold when it comes to manufacturing and distributing with no extra costs for the product software company (Cunsumano, 2004). On the other hand, disadvantages also exist, e.g. the requirements management of software is far more challenging with a higher release frequency for new product versions compared to non-software products.

In these last decades a shift in the global software market has emerged, i.e. a movement from customized software products to standardized software products (Weerd et al., 2006). It is because of this movement that the need for a new function has emerged; the software product manager. In contradictory to the rising interest and importance of SPM, not enough scientific or reference body of knowledge, literature or practical guidance exists regarding this subject (Ebert, 2009; Bekkers et al., 2010a). The consequences of practicing product management inadequately can impact the organization and business noticeably. Ebert (2009) has identified how poor product management can result in tangible problems by investigating the root causes. These tangible problems are:

- Wrong content
- Rework
- Delays, overhead
- Scope creep

These problems form a vicious circle: changes that need rework causes delays and challenges deadlines which in turn put pressure on the scope and so forth. The tangible problems can be spotted by early development project symptoms (Ebert, 2009):

- Conflicts of interest
- Unexpected dependencies between components
- Unclear cost/benefit
- Incoherent set of requirements.

These early project symptoms originate from root causes such as: vague vision and strategy, not integrated key stakeholders, unclear needs, not evaluated business case and unknown project boundaries. Consequences caused by these root causes are that customers are not satisfied with late products or products not satisfying their needs. This translates to a particular market not being fed what it needs, which has impact on the business, i.e. late product on the market equals discontent customers which equals decrease in sales turnover. Ebert (2009) states that these root causes are to be fixed, instead of trying to resolve the tangible problems or the early project symptoms of poor product management. Proper product management includes and guides activities such as portfolio management, product release life cycle and planning, requirements definition, product marketing and development, etc. These skills and competences adhere to the function of product manager and when executed properly equal the success of a product. Above all, assuring a winning business case is one of the most important aspects for the success of a product (Ebert, 2006).

3.1.1 Software Management Competence Model (SPM Competence model)

Software Product Management is a complex discipline and research area (Ebert, 2006; Weerd, 2009; Bekkers et al, 2010): it involves a great span of activities, immense amount of information to gather, analyze and make decisions upon, many responsibilities as a product manager and various stakeholders to take into account. These complexities drew Weerd et al. (2006a) towards the creation of a reference framework for SPM where the core is based on the software product in a hierarchical manner. Its purpose is to aid product managers with their daily practices. The reference

framework is the outcome of extensive literature studies as well as field studies (interviews and case studies) with primarily product managers at a major software vendor (Weerd et al, 2006a). This reference framework for SPM has been evaluated, which led to significant improvements and is reborn as The SPM Competence Model (Bekkers et al., 2010a).

As mentioned before, the structure of SPM Competence Model is presented in a hierarchical way, depicted in figure 4. On top, the Product Portfolio of the organization is represented and this is the complete collection of the products. For small organizations this is might be just one product, whereas larger organizations typically have multiple products usually due to product derivation. Each product in the portfolio can have various releases that are results of, for instance, bug fixes, new features, major architectural changes, etc. At the bottom, each product release consists of a combination of selected requirements. Each requirement adds to the functional or technical features of the product. Quality requirements such as performance, reliability or maintainability are also considered.

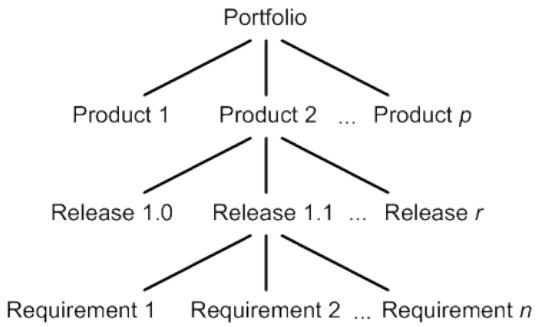

Figure 4: Artifact hierarchy of product management.

According to Dver (2003), adequate SPM is in fact a matter of organizing processes related to products, releases and requirements appropriately. The SPM Competence model (figure 1) gives an overview all the important processes areas in the SPM field. These important processes areas, called Business Functions, are Portfolio management, Product planning, Release planning and Requirements management. Each business function consists of several 'smaller' areas, called Focus Areas, of which each represents a strongly coherent group of capabilities. Another essential part of the SPM Competence model is the representation of the stakeholders, internal and external. Internal stakeholders are: Company board, departments of Sales, Marketing, Research & Innovation, Development, Support and services. Note that the Development department does not include development activities. However, Development serves as input for the SPM process areas. The external stakeholders are: the Market, Customers and Business partners. The arrows in the SPM Competence model indicate existing interaction between the different stakeholders, strong interactions between the adjacent business functions and the main flow of information and process between the focus areas. In the following sections, each business function with its corresponding focus areas is further explained.

3.1.1.1 Portfolio management

Portfolio management concerns the complete product portfolio of an organization and the strategic information gathering and decision making process thereof (Bekkers et al., 2010a; Weerd, 2009).
Cooper et al. (2001) define portfolio management as "a dynamic decision process, whereby a business's list of active new product (and Research & Developmet) projects is constantly up-dated and revised; new projects are evaluated, selected and prioritized; existing projects may be accelerated, killed or de-prioritized; and resources are allocated and reallocated to the active projects." In doing so, the risks of possible project failing is spread and projects are less vulnerable.

Furthermore, Cooper et al. (2001) explains that the ultimate goal of Portfolio management is to increase the value of the portfolio as a whole. The long-term success of product-oriented software organizations is highly dependent on effective portfolio management, i.e. the on-going process of defining, evaluating and prioritizing the set of existing and future product development activities (Vähäniitty, 2004). Various methods exist and are used in the industry to assess the current portfolio and possibilities to improve. However, the most popular portfolio methods yield the poorest results (Vähäniitty, 2004). For instance, financial methods (Return on Investment, Net Present Value, Break even, and more) are the most popular portfolio methods in the industry. This is not unexpected, since financial reasons are most important for proper portfolio management (Vähäniitty, 2004). However, financial portfolio methods do not yield the best results. As a consequence, organizations often use multiple portfolio methods in order to yield better, more trustworthy results.

The focus areas identified for Portfolio management are (Bekkers et al., 2010a; Weerd, 2009):

- Market analysis
  This focus area gathers market(s) information that is needed to support decision-making on the product portfolio of the software organization. When performing a market research it is important to define and focus on what you really want to know by means of a main research question. This main research question can be split in multiple sub-parts in order to make it easier to answer and more profound. Market research can be performed through different techniques. For instance*: Industry analysis* is dedicated research on a particular market or industry, *Political Economic Social Technological (PEST)* analysis is often used in combination with the SWOT analysis and is used to determine an organization's environment it operates in, *Value proposition* is an offer that presents the benefits a product promises to deliver in terms of quantity and *Product positioning* is a technique where products are compared with those of the competitors in order to know how each product is performing on the competing level.
- Product lifecycle management
  Information is gathered regarding products' life and significant changes. This information serves as input on key decision-making across the entire product portfolio such as the portfolio meeting the strategic business goals and needed changes to gain competitive advantage. More specifically, product lifecycle management deals with key-decision making from the initial conception of the product till it is phased out and has reached an end-of-life: decisions on which features to realize, decisions on release of products, modifications and improvements on products, diversification into new markets and phasing out products. The essence is to manage a product through its life cycle in order for it to become successful and brings in money for the organization. The typical life cycle of a product goes through the following stages: Initiate, Design, Build, Test & Integration, Release, Evolution and Phase out. Note that after the release of a product the life cycle is not completed. However, the stages Evolution and Phase out represent the products performance in the field that implies product evolution through various releases until it reaches a systematic end-of-life, i.e. Phase out stage. Another interesting way to look at product life cycle is from an economic interpretation, which includes the stages: Incubation, Growth, Maturity, Decline, End-of-life. This is a typical curve where costs in the first stage (after Incubation) should be minimized and revenue in the later phases (from Growth on) should be maximized. Note, this curve can deviate depending on the product and industry.
- Partnering & contracting
  The core of this focus area concerns with establishing partnerships, pricing models and distribution aspects in which the product manager plays a key role. Partnerships can be established with different types of partners*: implementation partners* install the software product by the customers and provide support or eventual trainings, *developing partners* develop product components, e.g. add-ons or plug-ins and *distribution partners* sell the software product. As a software product is an intangible asset (intellectual property), which consists of human knowledge, it needs to have a specific legal form for both protection and for the balance sheet as it is responsible for income. Intellectual property can have the following forms in a

software organization: *Copyright* is the exclusive right of the creator or obtainer to publish and distribute the work, *Trade Secret/Know how* concerns crucial knowledge related to business that can relate to products, processes, customers or way or working in the organization, *Trademark* is the exclusive right to utilize an expression of art or science in order to distinguish a character and establish a reputation and Patents is the exclusive right to exploit an invention and is issued by the national government authority.

3.1.1.2 Product planning

The core of Product planning is to gather the information for, and, the creation of the product roadmap, product line and/or its core assets as well as the process to manage all the product releases. (Bekkers et al, 2010a). Roadmapping, as it is referred to in software organizations, is traditionally called long-term product planning in the manufacturing industry and has been applied longer in this industry (Lehtola et al., 2009). However, the concept is less mature and researched in the software industry. Furthermore, roadmapping can be defined as a flexible technique that supports strategic and long-term planning and its goal is to investigate and communicate the linkage of markets, products and technologies over a period of time. Vähäniitty et al. (2002) describes roadmapping as "a popular metaphor for planning and portraying the use of scientific and technological resources, elements and their structural relationships over a period of time". In addition, the roadmapping process identifies, evaluates and selects strategic decisions to support achieving goals. The visual representation of roadmapping depicts the development, product release schedule, the supporting technology and the planned allocated resources (Vähäniitty et al., 2002). Rautiainen et al. (2006) identified three key values of long-term product planning (roadmapping), all of which should be addressed equally in order to gain the success of long-term planning: 1)Intent, supports coordinating complex activities; 2)Clarity, explanation of the direction of the Intent; 3) Awareness, supports short-term decision making and trade-offs.

The focus areas identified for Product planning are (Bekkers et al., 2010a; Weerd, 2009):

- Roadmap intelligence
  This focus area concerns the global information on markets, competitors and technologies, excluding requirements in Requirements management and in Release planning. The information should be presented in an abstract manner and is essential to the creation of the roadmap which supports decision-making by management. A wise starting point to gain intelligence for the roadmap is a Strengths, Weaknesses, Opportunities and Threats (SWOT) analysis (Weerd & Brinkkemper, 2010). The focus of a SWOT analysis is to identify connections on the points of each quadrant what are the best actions to take (Hill, 1997). Other types of intelligence also are of essence for the roadmap, such as Market intelligence, Society intelligence, Technology intelligence, Competitor intelligence and Partner intelligence.
- Product roadmapping
  Based on the information that is gathered, the actual roadmap is created. The organizations strategy and the product life cycle also serve as input. Roapmaps can be created for the short-term, up to two years, or for the long-term, three or more years (Weerd & Brinkkemper, 2010). According to Phaal et al.(2004), creating roadmaps can range from two extremes, namely technology push (divergent and looking for opportunities) and market pull (focused on customer defined product). The essence of the roadmap is to represent some type of a time-based chart, composed of various layers such as a product, commercial and technological perspective that is clear for management to make strategic decisions. An example of roadmapping is the technology roadmap which is serving as a technique to assist the planning and management of technology in relation to the other perspectives. This is critical since it assist in technological decision-making for management.
- Core asset roadmapping

Core assets are components that are shared by and in multiple products. This is usually the case with product lines (Northrop, 2002). Core asset roadmapping concerns mainly the planning of the development of existing core assets as well as future core assets (Bekkers et al, 2010a). Core assets imply a strong focus on reusability. This means a well-organized and transparent administration of core assets; systematic identification of core assets throughout the organizations products and main deliverables, registration and stored in a central location. This also makes it easier for maintenance of the core assets. In addition, core asset roadmapping can give insight on make-or-buy decision. Creating core assets roadmaps give insight on existing and future core assets, how these will evolve in time and provides information on product evolution. Examples of core assets are: software architecture, software driver, graphical user interface, business logic modules, test scripts, etc.

3.1.1.3 Release planning

This business function takes care of the process that creates and launches a product release successfully (Bekkers et al., 2010a). This means that Release planning manages the requirements set of each release in order to plan and launch the release. Van der Hoek (1997) defined Release planning as the process "through which software is made available to, and obtained by, its users". In market-driven software development, release planning is seen as one of the most critical tasks: selecting the right subset of requirements for implementation can mean the success of a software product (Carlshamre, 2002). Having a well-organized release planning process has added value for the organization. This helps keeping all stakeholders up-to-date on the future release, which implies an improvement for over all communication (Customers, Business partners, Development, etc.). It also makes decision-making less complex by communicating the right information to all stakeholders. The heartbeat principle clarifies the process of defining the release plan by sharing the knowledge (date and release in advance) of the release company wide and for the stakeholders. A heartbeat implies that the software organization has an agreed upon frequency on which releases are launched (Weerd & Brinkkemper, 2010), e.g. twice per year or once a quarter. Jansen & Brinkkemper (2006) distinguished different types of release updates, called update package. Update packages aims to improve the customers' current configuration and differ from updating the configuration intensively (major update such as bug fixes, new functionalities, architecture) to smaller updates (small updates such as bug fixes) and combination between those two.

The focus areas identified for Release planning are (Bekkers et al., 2010a; Weerd, 2009):

- Requirements prioritization
  Once all requirements are properly managed, the prioritization can start. Prioritizing the requirements points out candidates requirements that should be realized in the next product release and this is performed by the product manager and the (some of) other stakeholders. There are various requirement prioritizing methods, varying in ease-of-use that has as goal to identify the requirements that are of most value for the product. Organizations refer to this as the business value of a requirement. Every organization uses the prioritization methods that suits it best or its best practices. For instance, simple prioritization methods are MoSCoW (Ash, 2007) and Binary search list (Bebense et al., 2010) where requirements are prioritized by being granted a 'priority-identifier' or ranked by which requirements are then sorted or a binary search tree algorithm is used to rank the requirements, respectively. A more advanced method is for instance Integer Linear programming (Akker et al., 2005) where the required development time and/or effort and the estimated revenue of the requirements are considered for prioritizing. QUPER (Svensson et al., 2011) is an example of a more advanced method for prioritizing quality requirements by means of a cost-benefit analysis. Organizations often use some of these methods in combination with others for better results (Carlshsmre, 2002).
- Release definition
  The prioritized requirements (previous focus area) are selected, based on the priority they have

been assigned, and listed in the Release definition which will be used for implementation. The Release definition is a document that describes the to-be implemented requirements as well as the dependencies between those requirements of which the product manager is accountable for.

- Release definition validation
  The Release definition has to be validated before the actual realization is initiated. This is done by the people who will actually perform the realization; usually this is the Development department. The company board also has to approve the Release definition, based on existing product roadmaps and if the needed resources are available.
- Scope change management
  Scope changes can occur if for instance requirements are added or dropped, if resources are limited or new market opportunities present during development of the release. Scope change management takes care of the process that handles the scope changes that may appear during the realization of the new release. It serves as a monitoring process of the to-be realized requirements during development.
- Build validation
  Once the new release is realized, by the Development department, it has to be validated before it is made available to its users/customers. This procedure implies checking if all that was decided in the Release definition is actually realized.
- Launch preparation
  Finally, after the new release is validated it can be launched. This launch has to be communicated to involved stakeholders. Challenges that the organization has to face are for instance: communication of new product release features, new release documentation and necessary preparations for the implementation of the new product release.

3.1.1.4 Requirements management

Requirements management is the business function that is responsible for the complete and continuous management of requirements that are not yet included in a product release (Bekkers et al., 2010a). Regnell et al. (2005) state that this is the on-going process of handling the content and administrative data of each requirement, individually. In terms of activities, this translates to the gathering, identifying and revising and organizing of incoming requirements, which represents the focus areas of Requirements management. Whilst performing these activities, consideration needs to be taken with dependencies, core assets, themes and product lines. The sources where these requirements might originate are the internal and external stakeholders. The management of requirements can be complex: some organization can have large amount of requirements to process that comes from various stakeholders and complex requirement dependencies need to be managed.
Robertson & Robertson (1999) defines a requirement as “a statement on an action that the product is requested to do or a quality that the product is requested to have“. Furthermore, a distinction can be made amongst requirements types, i.e. functional requirements and quality requirements. A functional requirement describes what a product should do in particular situations or a service the product should provide (Sommerville, 2007). A quality requirement describes a quality (e.g. reaction time) of a product (part) or service that the product should have (Pohl, 2010).

The focus areas that form requirements management are (Bekker et al., 2010; Weerd et al., 2006):

- Requirements gathering
  This concerns the acquisition of requirements from both internal and external stakeholders such as customers, Sales, Development, Support, Research & Development and the company board. Techniques used to fulfill this activity are amongst others, Stakeholder interviews, User groups, techniques based on their advantages depending on the needs and the situation. Once gathered, the requirements are systematically stored.
- Requirements identification

After gathering and storing the raw requirements, the identification can start. Requirements are firstly separated from potential non-requirements. Next, duplicates are removed and requirements describing similar features are linked to each other. Usually, requirements originating from external stakeholders, e.g. customers or market requirements are often vague and not fully workable with for the organization. The organization at stake translates the market requirements into Product requirements for further processing. Furthermore, distinction is made between functional and quality requirements.

- Requirements organizing
  The organizing of requirements can occur in multiple ways, e.g. per core asset, per theme, per release or product for the entire life cycle of a requirement. The dependencies of the product requirements are substantial to continuously record and managed as necessary. For instance, some requirements require to be implemented first in order for other requirements to be implemented.

3.1.2 Software Product Management Maturity Matrix

The Situational Assessment Method for SPM (SAM-SPM) is proposed as an aid to assist product managers in improving their SPM practices and processes (Bekkers et al, 2010b). The SAM-SPM is used to assess an organization's current maturity level, determine possible improvement areas in order to increase the overall maturity level and evaluation on how to improve the method. In order to assess the organizations current maturity level, the method uses a maturity model based on the SPM Competence model structure to determine which capabilities are implemented and which capabilities should be implemented. By studying and analyzing what is implemented and what needs to be improved (organization's desires), gaps can be identified and incremental improvements can be suggested to the product manager.

Another important part of the SAM is the Situational Factors (SF) list. A Situational Factor can contain information on either process-level or on an organizational level. In addition, a Situational Factor describes the situational setting in which the SPM practices are performed and which has to be considered when improving SPM processes. For an elaborate explanation on the SAM-SPM we refer to Bekkers et al. (2010a).

The following four components complete the SAM-SPM:

- Knowledge base
  This contains the knowledge that will be used to build up the advice towards the product manager.
- Questionnaire
  Consists of two questionnaires: one for the implemented capabilities and one for the situational factors.
- Calculation
  Determines the current maturity, the optimal maturity and areas for improvements.
- Feedback
  Evaluation to update the knowledge base.

We focus on the Knowledge base, particularly the maturity matrix, since this is of key importance for this research. The SPM maturity matrix is important for determining the current state of organizations. The matrix is a Focus Area Maturity model (Steenbergen et al., 2010). A Focus Area Maturity Model provides incremental improvements, amongst other benefits. Each focus area has its own number of specific maturity levels (Bekkers et al., 2010a). Furthermore, the matrix represents the business functions, focus areas and capabilities in a best practice manner so it serves as a guideline for software organizations. Another sub-component of Knowledge base is the Situational Factors (SF). Situational Factors represents information of a process, context of the organization and

the organizations itself (Bekkers et al., 2008). Their purpose is to help determining the current situation of the organization. Regarding the SPM maturity matrix, SF's explains the situational context in which the SPM processes occur and would need to be improved as well as in which the product manager has to operate in. In the SPM Maturity matrix (see table 4), the business functions and focus areas are represented in the leftmost column of the table. The maturity levels of the focus areas are represented with the letter A through F (capabilities) and range from maturity level 1 to 10. The letters (A - F) represents capabilities, e.g. for focus area *Requirements gathering*, capability *A* is defined as *Basic registration*. The maturity level is determined by the highest level of a capability, before a capability has not been satisfied by the organization. Table 5 gives an example of how a capability is defined and its attributes.

Table 5: Software Product Management Maturity Matrix.

Maturity level / Focus area	0	1	2	3	4	5	6	7	8	9	10
Requirements Management											
Requirements gathering		A		B	C		D	E	F		
Requirements identification			A			B		C			D
Requirements organizing				A		B		C			
Release planning											
Requirements prioritization			A		B	C	D			E	
Release definition			A	B	C				D		E
Release definition validation					A			B		C	
Scope change management				A		B		C		D	
Build validation					A			B		C	
Launch preparation		A		B		C	D		E		F
Product planning											
Roadmap intelligence				A		B	C		D	E	
Core asset roadmapping					A		B		C		D
Product roadmapping			A	B			C	D		E	
Portfolio management											
Market analysis					A		B	C	D		E
Partnering & contracting						A	B		C	D	E
Product lifecycle management					A	B			C	D	E

Table 6: SPM capability example; Requirements organizing: A.

A	*Requirements organization*
Goal:	Increase potential of requirements by identifying value outside of the original boundaries, and provide insight into the planning concerning the requirement.
Action:	Product requirements are organized based on shared aspects (e.g. type, function, or core asset).
Prerequisite(s):	Requirements gathering A

3.2 Software Product Lines

In this section, we discuss the subject of Software Product Lines (SPL). The aim is to present SPL literature that relates to software product management. Currently, existing literature that describes SPL which also take SPM into consideration are somewhat limited. Existing literature mostly discuss more product *development* related than product *management* related. First, we elaborate more on SPL in general. The sub-sections that follow will discuss SPL in-depth, as result from the literature study.

Software product lines can help a software company excel significantly and in various ways: less development and maintenance costs, faster time-to-market, improved product quality, improved

customer satisfaction, reuse of artifacts such as architecture, drivers, source code, requirements and more (Northrop, 2002; Bosch, 2002; van der Linden, 2002; Birk et al., 2003; Pohl et al., 2001b, Clements, 2005; Böckle et al., 2005). However, these advantages claim a remarkable amount of effort. Bosch (2002) states that companies that are interested in employing SPL need to consciously and explicitly consider the change of software development to a SPL approach. The maturity of the company as a whole also plays an important role. The more mature the company is in its management, domain understanding and project organization, the less effort is needed to adopt a SPL approach.

Bosch (2002) presented a matrix relating SPL approaches, SPL artifacts and maturity levels and the organizational structures software organizations can adopt. The columns (table 7) represent the SPL approaches (SI to PPL). The first nine rows represent maturity levels of the artifacts and the last four rows represent the organizational models. In the matrix, a '+' or a '+/-' represents combinations that work well together, sort of best practices.
The absence of a '+' or a '+/-' does not directly mean incompatibility. These combinations require additional effort and resources to achieve according to the research of Bosch (2002) or are less common in practice. The matrix is presented below.

Table 7: Relating SPL approaches to SPL artifacts and organizational models (Bosch, 2002)

Artifacts and Organization	Product Line Approaches					
	SI	P	SPL	CPB	PP	PPL
Under-specified Architecture	+	+			+	
Specified Architecture			+		+	+
Enforced Architecture				+		
Specified Component	+/-	+			+/-	
Multiple Component Implementations		+/-	+		+	+/-
Configurable Component			+/-	+		+
Architecture Conformance	+					
Platform-Based Product		+	+		+	
Configurable Product Base				+		+
Development Department	+	+	+			
Business Units	+	+	+			
Domain-engineering Unit			+	+	+	+
Hierarchical Domain-Engineering Units					+	+

Bosch (2002) identified six SPL-approaches that aim to reuse in an architecture-centric and intra-organizational manner. These approaches take various forms, i.e. ranging from simple systems development to more large and comprehensive systems development and are organized in a number of levels. The maturity development path describes the levels below:

- Standardized infrastructure (SI)
  This approach provides the first step towards software artifacts reuse. Namely, it focuses on standardizing the infrastructure on which the future products will be based on, which consists of the operating system with the typical commercial components such as a GUI or database management system.
- Platform (P)
  Here, a platform is developed on which the products and applications are based on. This is on top of a standardized infrastructure. The platform is responsible for the total commonalities of the products and applications.
- Software product line (SPL)
  A platform is extended (e.g. in functionality) to the point that functionalities that are common in most products are included in the shared artifacts. This is called a Software Product line. Product(s)-specific functionalities still exist and are part of the product deviation.

- Configurable product base (CPB)
  This approach is applicable when organizations operate in relatively stable domains with a large product orders. These organizations prevent developing different products and instead move towards one configurable product base where the product is configured into the product bought by the customer, either at the company or at the customer site.
- Program of product lines (PPL)
  This is an approach that is made up of a software architecture that is defined for the whole system and the components that the system is made of. Most of the components configuration that results in a system is SPL's. These can practically be configured as the Configurable product base approach or through SPL-based product derivation.
- Product population (PP)
  Product population extends the amounts of products that can be derived from the shared product line artifacts. This refers to the situation where the existing sets of functionalities are extended so a more diverse set of products can be derived.

Next to the SPL approaches, three types of SPL artifacts have been identified: SPL architecture, shared components and products derived from the shared artifacts. For each of these artifacts, three levels of maturity have been described, similar to the maturity levels described above. These maturity levels are presented in the matrix (first nine rows).
The primary organizational models that can be applied when implementing a SPL approach are (for a detail explanation on the both of the maturity levels or the organizational models, consult Bosch (2001, 2002)):

- Development department
  In this model, the employees are considered as a resource that can be assigned to different project, i.e. domain engineering project or application engineering project. Thus, no specialized organization model is needed when all development is taken place at one department.
- Business units
  A Business unit is dedicated on the type of products. Each business unit is responsible for the development and evolution of a subset of products or one product in the product line. All the business units share the reusable assets in the product line.
- Domain-engineering units
  Traditional literature suggests this as the organizational model for SPL. The domain-engineering unit is responsible for developing and evolving new or existing reusable artifacts that were mentioned earlier.
- Hierarchical domain-engineering units
  This typical organizational model is needed where hierarchical products exists. In these cases, a domain engineering-unit exists to develop reusable assets to be used in another lower domain engineering-unit that will develop the end product.

Previous work of the Software Engineering Institute (SEI) confirmed that for organizations to succeed with a SPL approach the organization must be willing to alter its technical and management practices as well as the working organizational structure, personnel and business approach (Northrop, 2002). In addition, SEI defined a software product line as:

'a set of software-intensive systems that share a common, managed feature set satisfying a particular market segment's specific needs or mission and that are developed from a common set of core assets in a prescribed way'.

In practice this comes down to taking applicable components (e.g. architecture, source code, software modules, drivers, requirements, documents, etc.) from a common, shared asset base, after which the components are tailored through preplanned variation techniques as needed, any new components are added if necessary and assembling the final product according to rules of the

common, product-line wide architecture. The use of common assets in order to develop products requires planning, investment and strategic mindset that focus beyond the boundaries of a single product.

Different organizations worldwide use their own implementation of a SPL approach. Fortunately, SEI has distilled the global and vital activities of such approaches. At the highest level of abstraction, three extremely iterative activities can be identified that intertwine technology and business practices (Northrop, 2002). These activities are *Core asset development* and *Product development* which are guided by technical and organizational *Management*. Traditionally, the activities Core asset development and Product development are called Domain Engineering and Application Engineering, respectively (Böckle et al., 1998; Clements, 1999; Weis et al., 1999; Northrop, 2002; Pohl et al., 2001b; van der Linden, 2002). These activities are widely accepted in software product line literature as the core processes necessary for implementing a SPL development approach. These will be elaborated in-depth in the coming sections.
Given the definition on SPL by Northrop (2002), the aim of Core asset development is to establish production capability by means of developing core assets. Product development's aim is to assemble the products out of the core assets. However, these activities are bi-directional, i.e. new or revised core assets often evolve from Product development, i.e. existing or completed products (SEI, 1999; Northrop, 2002).

Böckle (1998) defined Software Product Line Engineering as "a paradigm to develop software-intensive systems and software products using platforms and mass customization". In contrary to standard products (non-individualized products), mass customization aims to supply the demand for large-scale production of individualized products, implying taking the customers' requirements into account and providing the product they want (Böckle, 1998). A platform supports mass customization by providing, and the ability to develop, common parts that will be needed in the final product, i.e. a collection of reusable artifacts. Thus a software platform can be defined as

'a set of software sub-systems (source code, requirements, architecture, test plans, and other artifacts) that form a common structure from which a set of derivative products can be efficiently developed" (Meyer et al., 1997)'.

Furthermore, the common parts used to develop multiple products have to be sufficiently versatile in order to fit the to-be developed systems. This versatility facilitates mass customization (versatile artifacts are reusable in different systems) and is refer to as Variability in the SPL context (Pohl, et al., 1998). Variability in reusable artifacts implies the commonalities and especially the differences across these artifacts. Pohl et al.(1998) state that the systematic combination of a platform and mass customization as a development approach for software-intensive systems is the core of SPL engineering. This leads to Domain and Application engineering and thus also Core asset and Product development as defined by SEI (1999).

Implementing a software product line approach is not a decision made overnight. As a software company grows its product(s) will enhance and it will desire to provide tailored products at reasonable costs in order to satisfy the market. However, product management becomes more and more challenging, yet more important. A SPL approach involves clear changes in product development and management, organizational structures and management support which are also not implemented overnight. In the following section we will discuss the most essential SPL-literature according to most researches with respects to SPM.

3.2.1 Domain and Application engineering

Software product line engineering separates two main processes: **Domain engineering** (hereafter referred to as DE) & **Application engineering** (hereafter referred to as AE). Weis et al. (1999) presented a SPL engineering framework based on the different aspects of the two main processes in which they are modeled. This framework is presented in figure 5.

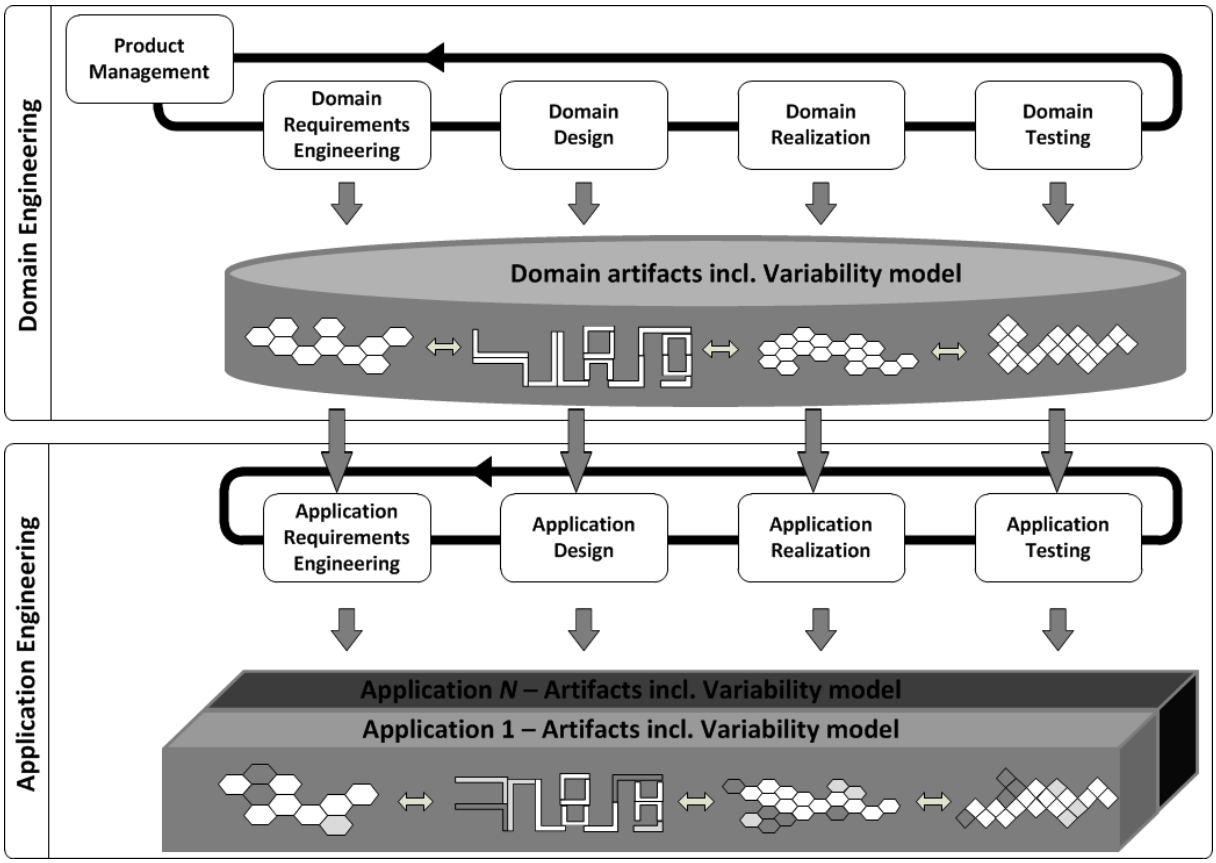

Figure 5: The Software product line engineering framework (Weis et al., 1999)

Both processes develop artifacts: domain artifacts and application artifacts. The **domain artifacts** are **reusable** and are developed from the sub-processes of domain engineering. These form the platform of the product line and serves as input for the specific applications development. The **application artifacts** represent part of the **tailored** product line applications. As the product line produce multiple products, application engineering is responsible to manage the product-specific artifacts for each product separately. Next to defining the variability of the SPL, DE aims also to define the scope of the SPL (the products the SPL is destined for) and develop the reusable artifacts that fit the necessary variability (Böckle et al., 1998). DE consists of the following sub-processes (Pohl et al., 1998):

- Product Management
  This sub-process's main concern is the management of the product portfolio of the software organization as it will be expanded by products originating from the SPL. Existing products or artifacts are listed to be reused for development of the platform. Through scoping techniques, the scope of the SPL is defined. Top management defines the strategic goals which serve as input for product management. Product management translates the input into a product roadmap that determines the key common and variable product features of the future products as well as their release planning. Product management for single systems differs from SPL (Böckle et al., 1998). Firstly, the introduction (or elimination) of a platform has a strategic

meaning for the organization, with respect to business success. This gives the organization the opportunity to provide various product variants at reasonable costs. Secondly, the products in the portfolio, originating from a platform, closely relates to each other, compared to single systems that can differentiate immensely from one another. Thirdly, product management pays close attention to the evolution of the market(s) and technology, customers' needs, standards, modifications in legal constraints, product features; all for future products to- be developed.

- Domain Requirements Engineering (DRE)
  The identification and documenting of common and variable requirements for the SPL is the main concern here. The product roadmap of the previous sub-process is the input for this one. Domain requirements engineering outputs reusable, textual and modeled requirements and also the variability model of the SPL. The domain variability model defines the variability of the SPL, i.e. where products vary (variation points), how they will vary (variants) and dependencies which have to be considered. The requirements are analyzed whether they are common for all products or specific for products amongst the rest (variability). The abstraction of the variable requirements is modeled in the variability model of the SPL. Based on the product roadmap, this sub-process also foresees changes in legal standard, markets, features, technology that influences requirements for future products or applications. The last three statements are different when single systems are dealt with (Böckle et al., 1998).
- Domain Design
  This sub-process involves all necessary activities that define the SPL reference architecture, which provides an overall, high-level structure applicable for all the product line applications or products. The domain requirements and the variability model of the previous sub-process serve as input for design of the reference architecture. The output consists of the reference SPL architecture and refinement of the variability model, i.e. including variability that is necessary for technical reasons (internal variability). In difference with single systems, domain design incorporates flexibility from the very beginning to support variability of the SPL and the reference architecture can be modified according to the requirements of the to-be developed applications. Domain Design indicates both the reusable components that are developed and tested by DE as the product-specific components developed and tested by AE (Böckle et al., 1998).
- Domain Realization
  This sub-process handles mainly the realization of the reusable software components and with the detail design, prior to the realization. The SPL reference architecture and a list of the to-be developed reusable artifacts serve as input. The output includes the detailed design and implemented artifacts of the reusable components. Compared to single systems, domain realization delivers components that are loosely coupled and configurable, instead of a running system. In addition, the components are planned, designed and developed for reuse in the different products or applications. Domain Realization integrates configuration mechanisms into the reusable components. This is necessary for the realization of variability in the SPL.
- Domain Testing
  Domain Testing validates and verifies the reusable components according to their specification; the requirements, the reference architecture, design artifacts and the developed components, which also serve as the input. This sub-process also takes care of reusable test artifacts for application testing in order to reduce effort. The output is simply the test results as well as reusable test artifacts. Remarkably, there is no running application to be tested. These would be available in application testing.

The lower part of figure 3 shows AE and its sub-processes. AE aims to reuse as most as possible when developing the specific SPL products or applications by exploiting the **commonalities** and the **variabilities** built in by DE. The AE artifacts are documented and related to DE artifacts. This has the purpose to be able to trace where reusable components get used for. In AE, the variability is used and bound according to the needs from the DE artifacts (requirements, architecture, components

and tests). This variability is of key importance for making a SPL successful, since it gives the ability to differentiate easily and as needed. AE consists of these sub-processes:

- Application Requirements Engineering (ARE)
  This sub-process concerns the needed activities for developing the application requirements specification. Depending on these requirements, the reuse of the DE artifacts can excel greatly or not. This is a challenge for application requirements engineering, which have to detect differences in application requirements and what is available from the platform (DE artifacts). This implies that the platform would need to cover the needs of application requirements as much as possible. If not, then AE initiates the development of another reusable component. The domain requirements and the roadmap with product features corresponding to the product or application are inputs. Here, specific customer requirements may be added that were not collected in domain requirements engineering. The output is the specific product or application requirements specification. Compared to single systems where requirements are usually newly added, most of the requirements here derived from domain requirements and is based on the communication of the available commonality and variability (Böckle et al., 1998). During this sub-process the difference (delta) between domain requirements and application requirements must be identified and evaluated against the amount of eventual adaption effort needed and documented properly.
- Application Design
  The core focus of application design is the activities needed for the application architecture. For this the SPL reference architecture, which is the input, is used to instantiate the application architecture. Particular parts of the SPL reference architecture are selected and configured into application specific adaptations. Thus, the output exists of the application architecture for the specific product or application. Binding variability (making choices where the reference architecture gives variants to do so) makes it possible for the application architecture to get deducted from the SPL reference architecture, instead of developing a completely new architecture. When doing this, Application Design has to adhere to the rules of binding the variability of the reference architecture (variability dependencies). Structural changes that would require effort equal to developing from scratch must be rejected as adaption effort, with respect to the deltas.
- Application Realization
  This sub-process concerns the development of the particular product or application. Mainly, this involves the selection and configuration of reusable software components provided by DE. Application-specific assets are also realized and together with the reusable assets these form the product or application. As input, the application architecture and the reusable artifacts from the platform are used. The output is the workable or running product or application with the detailed design artifacts. When compared to single systems, many of the software assets (components, requirements, interfaces) are not newly developed. However, they are derived from the platform by binding variability. Application-specific realization is possible; however it must fit into the reusable artifacts.
- Application Testing
  This sub-process involves the activities needed to validate and verify the product or application against its specification. The inputs for this are all the application artifacts mentioned before (requirements, specification, architecture, components, and interfaces), the realized product or application and the test artifacts provided by domain testing. The output is simply test results of all the performed tests. The detected defects or necessary changes are documented properly in problem reports. The used test artifacts are not developed newly; they are derived from the platform. Additional test are performed in order to test the configurations and that the right variants have been used.

The table below gives an overview of the sub-processes of DE and AE and their corresponding artifacts. **Domain** artifacts constitute the **platform** of the SPL and are all stored in a central

repository. These artifacts are interconnected by traceable connections in order to keep the specification of the commonality and the variability consistent among the artifacts. **Application** artifacts include all development artifacts for a **specific product** or application and the configured, tested and running product or application itself. Applications assets are also interconnected by traceable connections in order to have a correct binding of variability among the artifacts. Many of the application artifacts are specific instances of the reusable domain artifacts. The traceable connections are also used for the SPL evolution, i.e. application artifacts that are influenced by modifications of domain artifacts can be easily determined.

With respect to SPM, it is clear that the product management process coordinates DE which proves that product management has great responsibility. Some of the artifacts, DE as well as AE, can be clearly related to the SPM Competence model, e.g. roadmap and requirements specification. However, artifacts such as reference architecture or variability model are not discussed in the SPM Competence model, whilst still being part of product management, product management for SPL. Thus, there are specific product management practices for SPL.

Table 8: Artifacts of Domain and Application Engineering (Pohl et al., 1998).

DE sub-process	Domain artifact
Product management	Roadmap (+planning future release dates)
Domain Requirements Engineering	Requirements (textual and modeled) and Variability Model
Domain Design	Reference architecture, refined variability model (including internal variability)
Domain Realization	Detail design models, implementation artifacts (source codes, configuration files, make files, etc.)
Domain Testing	Test plans, Domain test cases, domain test scenarios.
AE sub-process	**Application artifact (application assets)**
Application Requirements Engineering	Application Requirements Specification and Application Variability Model
Application Design	Application Architecture
Application Realization	Detailed design artifacts (component and interface), (configured) running application or workable product
Application Testing	Application test documentation, problem reports

Core asset development, Product development and Management

As mentioned previously, SEI (Clements, 1999; Northrop 2002) also describes a SPL development approach based on their research. It is similar to the DE and AE described above, apart from the fact that SEI has added the process of Management to the approach. This practice's main focus is (also) on developing core assets (reusable components) and building products from those core assets under the supervision of Organizational and Technical Management. **Core asset** development (Domain Engineering) sets the first step towards establishing the ability to initiate production. **Product development** (Application Engineering) turns out products from the core assets. However, these two processes are highly iterative, between and within. This acts as a feedback loop between core assets and products; core assets are refined or even newly created as the organization develops products. **Management** has great contribution in the success of the SPL and must therefore be strongly committed. Technical management supervises the activities of core asset and product development, ensuring that the involved personnel undertake the required activities, follow the defined processes for the SPL and gather data to trace progress. **Organizational management** is responsible for setting up the proper organizational structure that best fits the organization and ensures the organizational units receive the right amount of resources (e.g. personnel). Both core asset development and product development has inputs and outputs. For core asset development these are:

- Product constraints

Commonalities and variations among the to-be developed products (of the SPL) and their features.

- Styles, patterns and frameworks
  These are relevant architectural aspects necessary when defining the architecture taking into concern the product and the production constraints.
- Production constraints
  Standards and requirements from different stakeholders that apply to the products in the SPL.
- Production strategy
  This is the general approach for realizing the core assets. Either core assets are developed or from these products or products components are generalized and used to develop core assets. Usually a combination of both approaches take place.
- Inventory of preexisting assets
  Available assets, software or organizational, that can be added to the assets base for reuse.

The output of core asset development also serves as input for Product development with the addition of individual product requirement (this is equal to application specific requirements):

- Requirements
  Individual product requirements.
- Production line scope
  Defines the products that the SPL will be capable of developing. This involves the commonality and variability of each product member of the SPL.
- Core assets
  Reusable components that form the basis of the products originating from the SPL sharing an overall architecture. Next to software components these can also be documentation, test plans, integration plans and all design components. These are considered as supporting artifacts
- Production plan
  Describes how the products are developed from the core assets.

The similarities with DE and AE are clear. The basic idea of developing reusable artifacts and using these to build the end-product or application is the same. The input for core asset for instance, also includes product specification, defining architecture for products to adhere to, existing assets that will be used as input for product development. The core assets are developed according to an overall architecture and so are additional components such as, test plans, integrations plans and all sorts of design documentation. Unfortunately, the authors of SEI did not describe the inputs and outputs as elaborately as the authors of DE and AE. However, these two researches shows the importance of separating processes that are focused on the core assets and processes that are focused on developing the products (mainly) using the core assets.

3.2.2 Business, Architecture, Process, and Organization (BAPO)

This practice recognizes the processes of **Domain** and **Application Engineering** as the development approach of for SPL. When implementing a SPL development approach four concerns are taken into account: **Business (B), Architecture (A), Process (P)** and **Organization (O)** (Linden, 2002).
Business deals mainly with the scoping (product line, domain and assets) of the SPL which has great impact on the business, i.e. earning profit from the products the SPL is able to provide. **Architecture** deals mainly with the technical specification of the product line needed for realization, i.e. significant requirements, reusable components, concepts, design, structure, texture, tests. Important is that the architecture can deal with commonality as well as variability. Here variability management plays a vital role. Requirements modeling has a clear link to Requirements management from the SPM Competence model (Requirements organizing, traceability), i.e. ability to trace requirements through the sub-processes knowing in which asset they are used. **Process** deals with the software development process that reuse assets in order to build products. This is the domain development

process (domain analysis, design and implementation) and the application development process (application requirements, design and coding). These are described merely differently than DE and AE by Pohl et al. (2005), however the essence is very similar. **Organization** deals with the organizational structuring and influence a product family development approach (SPL) has on an organizational level.

These 4 interdependent concerns (BAPO) are used to provide 4 dimensions of the software product family evaluation framework. Each concern has **evaluation levels** that may be applicable for an organization, which are influence by multiple aspects.
Business' evaluation levels are: *Reactive, Extrapolate and Proactive*. These are influenced by the organizations: *Identity, Vision, Objectives and Strategic Planning*. With each Business' evaluation level the aspects get clearer, more developed and better managed. E.g. at the Reactive level Requirement Management is mostly ad hoc while it is better planned and managed in the succeeding levels. **Architecture's evaluation** levels are: *Independent Product Development, Standardized Infrastructure, Software Platform, Software Product Family* and *Configurable Product base*. These are influenced by: *Product family architecture, Product quality, Reuse level, Domain* and *Software Variability Management*. With every increasing level the overall architecture aspects evolve into more specified architecture models, increasing reuse levels, increasing quality and the domain would be well managed and/or established.
Process' evaluation levels are: *Initial, Managed, Defined, Quantitatively managed* and *Optimizing*. These are influenced by: *Predictability, Repeatability* and *Quantifiability*. From the first level through to the last, development will become more predictable, the development process will become more repeatable and more data will be available to quantify the development.
Organization's levels are: *Unit oriented, Business lines oriented, Business groups/divisions, Inter-division/companies* and *Open business*. These are influenced by: *Structure, Culture and Roles & Responsibilities*. With every level the structure of the organization gets more complex and less informal, the culture gets more focused, cooperative and competitive and the roles & responsibilities will become more specialized on product development. A representation of the evaluation framework would be a target evaluation profile. The authors acknowledge this; however they did not work that part out (beyond paper scope). A company input profile, including domain type and business strategy are mentioned as being necessary for the profile. The purpose of the profile is similar with that for the SPM maturity matrix, with incremental improvement possibilities through the situational factors.

3.2.3 Variability management

Bosch (2000) states that the dependencies in the various products of a product line makes the evolution of the SPL more complicated compared to stand alone products. In addition, the possibility of conflicting requirements between the various products makes things even more challenging. Variability is described as '*the ability to change or customize a system*'. Improving variability in a system lets the system adapt easier to changes. By using Feature Graph Notation (Feature Modeling) variability is identified, as well as commonality, and is clearly presented and modeled. In this work, a feature is considered as abstraction from related requirements, indicating that specifying requirements (RM) is vital for the feature graph. The proposed method of managing variability consists of the following steps:

1. Identification.
   The feature graph notation is used to identify where variability exists or must be implemented. With this diagram, the variation points can be determined.
2. Constraining variability.
   In this step detail actions are taken that will constrain the variation points, allowing just enough flexibility. This is accomplished by activities such as choosing the binding time for variation points,

addition of variants, setting variability pattern for variation points and choosing representation of the variation points.

3. Implementation.
   Here a suitable technique needs to be chosen for realization.
4. Managing the variants.
   Based on the variation points, variants may be added, manually or automatically by means of system automatic updates for instance.

With respect to previous sections, Lauenroth et al. (2005) state that variability modeling is used to model variability of DE artifacts, where variability is firstly defined. In AE variability is exploited through binding the defined variants. Variability management is a technique that introduce and manages the flexibility needed in a SPL. The flexibility gets incorporated in artifacts and in turn is realized through the various products or application that the SPL can turn out. Thus, it is an essential activity with respect to the steps prior to, and, development.
The strongest relation it has with the SPM Competence model is with Requirements management, i.e. in the earliest phases of DE, common and variable features of the SPL products are specified. In the following sub-processes, variability is specified more in details through requirements, design, realization and tests.

3.2.4 RequiLine

RequiLine is a tool developed to support requirements engineering for SPL (von der Maßen et al., 2004). The task and challenge of RE for SPL is the elicitation of requirements that are shared by all products in the product line and requirements that are specific to certain products. These requirements are usually in great amount, are mandatory or variable, have interactions and dependency with other requirements and all this has to be managed properly. RequiLine also supports Feature modeling, since variability (variable requirements) is one of the most essential characteristics of SPL and needs to be modeled and managed. One requirement for the tool was to be able to manage requirements as well as their attributes, support information and dependencies that have been stated for the features. This implies that features can be linked to requirements and vice versa. This is a clear example of Requirements Identification and Organizing of the SPM Competence model. Generally, researchers on the topic of SPL often find there is a lack of tool-support for SPL-development.

3.2.5 Release planning for product lines architecture

Taborda (2004) proposed a release matrix as a mechanism to facilitate the planning, communication and coordination of **incremental releases** by combining traditional **Requirement Engineering** and **Configuration Management** principles. The matrix takes two distinct management views into account: **Products** and **Components.** In the x-axis the components are presented while in the y-axis the products, that use the components, are presented. The matrix records a relationship between product (P*i*) and component (C*j*) in the intersecting cell (R*ij*). When no relationship exists a null entry is presented in the cell. The content of each cell can be considered as the scheduled dates of the set of dependent releases. In addition, multiple matrices can be used in order to record different life-cycle data. Each row of the Release Matrix represents a product's release plan that originated from and must also be compatible with the component's release that the product is dependent of. Likewise, each column represents a component's release plan that is based on the total set of the product requirements that need to be implemented in that release. This practice indicates the use of **release planning** in SPL. A clear difference is that not only requirements are planned for future releases, however requirements are 'bundled' into components and components can form product features, and these are planned for future releases.

3.2.6 Product Line Portfolio Planning using Quality Function Deployment (PPP-QFD)

Product Portfolio Planning (PPP) is closely associated with product development. However, it is a management activity (Helferich et al., 2005). PPP has the task to produce and manage a portfolio of products that will optimally satisfy customer demands while restricting the total number of products offered (Helferich et al., 2005). As for product lines, portfolio planning address topics such as product line members, commonalities, variations, technology utilization and product line evolution. This description of Portfolio Planning is similar with that of Portfolio management given by the SPM Competence model. Quality Function Deployment (QFD), which is used to identify true customer needs and features, gives a systematic way of communicating between customers and developers in a yet informal way. The proposed approach of PPP makes extensive use of QFD by means of the best known instrument for QFD, House of Quality (HoQ). HoQ is a matrix which analyzes customers' requirements in detail and translates these to developers' understanding. In general, this is done by:

1. Collecting requirements for the product line through existing and potential customers which are then processed (analyze, sort, prioritize, etc.).
2. Developers, software architects and selected customers are brought together to build the HoQ.
3. Developers and architects evaluate and analyze different possibilities for software architecture and technologies to be used taking the quality attributes and product functions into account.
4. The result of the previous step is used to build prototypes to present to the customers.

Requirement management, as described by the SPM Competence model, is somewhat similar to this approach, i.e. when requirements need to be gathered for the product line. This indicates not much difference between RM of a single product or a product line. The focus areas of Market analysis and Product Lifecycle management are key points in the PPP. The downside of this approach is that is has not been validated in the industry yet.

3.2.7 Requirement-based taxonomy for SPL evolution

Similar to most of the SPL-literature presented previously, this one acknowledges two main cycles in SPL development: Domain and Application Engineering. This SPL practice proposes a taxonomy as *'means for categorizing requirements changes in a product line context'* (Schmid et al., 2007). The evolution of requirements in SPL can happen on three levels:

- **Requirements level change** (changes to individual or small group of requirements)
- **Product level change** (changes to products)
- **Product Line level change** (changes to whole product-groups).

In addition, the categorization of the requirements is:

- **Commonalities** (requirements that are common to all products in the SPL)
- **Variabilities** (requirements that are not common for all products)
- **Product-Specific** (requirements that are only relevant to an individual product).

Usually multiple products would include requirements from all three categories. The changes on the three types of level can bring the following actions:

- **Requirements level change**
  Adding, Deleting or Modifying individual or groups of requirements. These changes can happen on all three levels described above.
- **Product level change**
  Additions and Deletions of whole products. Modification of individual product is considered to be changes on requirements level. Adding a product implies specification of variabilities and product-specific aspects. Deletion implies also the deletion of product-specific aspects.
- **Product line level changes**: Adding, Removing, Merging and Splitting of a product line.

The implications of the different level changes are elaborated together with the influence of the evolution actions has on traceability information. However, this taxonomy has not been put into practice. The authors figure that capturing the complete set of possible level changes with the complete set of actions, that that would be enough.

3.2.8 PuLSE

PuLSE is a methodology that makes the conception and the deployment of SPL within large variety of enterprise contexts possible. The main elements that compromise PuLSE are phases and components: **Deployments phases**, **Technical Components** and **Support Components** (Bayer et al., 1999).

The *Deployment phases* describe the logical stages and activities performed to set up and deploy the product line. This is realized by base lining and customizing the methodology according to the enterprise (*Initialization*), scoping, modeling and architecting SPL infrastructure (*Infrastructure Construction*), using the infrastructure to develop SPL members or components (*Infrastructure Usage*) and evolving and managing the SPL over time (*Evolution and Management*).

The *Technical Components* provides the technical knowledge that is needed to operationalize the SPL development. Facets of different Technical Components are used in each of the developments phases. These are specified for: know-how for the *Initialization Phase*, know-how for product scoping, modeling based on product characteristics and architecting the reference PL architecture for the *Construction Phase*, know-how for performing the *Usage Phase* and know-how for configuration management over time for the *Evolution and Management Phase.*

The *Support Components* are the bundles of information or guidelines which provides a better adaption, deployment and evolution of the SPL. These components are used by the other two elements: *Project Entry Points* (customization of PuLSE for major projects), *Maturity Scale* (integration and evolution path for the SPL adoption) and *Organization Issue* (guidelines for the appropriate organizational structure set-up).

Although PuLSE gives a complete methodology for developing a product line, it pays less attention on SPM aspects as defined by the SPM Competence model. In the last phase, Usage, customer requirements are used in order to plan and develop a new product line member. This is a mere example of the essence of Requirement management and Release planning. However, in the same phase product line members are specified, instantiated and validated which can be related to Product Lifecycle Management. Unfortunately, these topics are not addressed elaborately.

3.2.9 Product Derivation framework

This practice also considers a SPL approach to be comprised of a two-staged process: Domain and Application Engineering. However, this practice focuses on the **product derivation process** that occurs during Application Engineering (Deelstra et al., 2004). Deelstra et al. (2004) states that product families can be classified into two scope dimensions, i.e. **Scope of reuse and Domain scope**. Scope of reuse refers to the extent to which the commonalities between related products are exploited. Domain scope refers to the extent of the domain(s) in which the product family is applied.

The Product Derivation Process is based on the scope of single product family (single product line) which is used to derive multiple related products (Deelstra et al., 2004). The process consists of two phases: **the initial phase** and **the iteration phase**. On top of these two phases, **Requirements Engineering** manages the requirements throughout the entire process of derivation.

In the initial phase a first configuration is created from the assets in the product line. In this phase, two approaches for deriving the first product configuration can be used: Assembly (assembly of a subset of shared assets into the first product configuration) and Configuration selection (selecting the closest matching existing configuration available).

In the iteration phase, the first configuration is modified in a number of subsequent iterations until the product adheres to the imposed requirements. In this phase, the steps Modification and

Validation are central. Modification is applied by selecting architectural components variants, different components implementations variants or modifying parameters settings. These represent the three abstraction levels (architecture, component and parameter) on which modification can happen. In Validation the system is validated in order to assure that it adheres to the requirements and consistency and correctness of the components configuration is checked. If the initial configuration properly adheres to the requirements after the initial phase, then the product is finished. However, if this is not the case, the iteration phase is initiated.

3.2.10 SPL FAST Process

The Family-oriented Abstraction, Specification and Translation (FAST) process can be seen as an alternative to traditional software development process. Ardis et al. (2000) developed a systematic process that is applicable when organizations develop multiple versions of a software product that share significant common attributes such as behavior, interfaces and source code. FAST also recognizes the two main processes of **Domain** and **Application** Engineering as the environment where SPL development takes place. The **common** and **variable** characteristics (features) of the product family are identified in a Commonality analysis. This analysis is documented in natural language and also registers (including illustrations) the scope, anticipated issues and terminology of the product family. In order to distinguish further the differences between the family members it is common practice (in the FAST process) to make use of **example scenarios**. **Usability scenarios** describe '*the actions required to perform common user operations*'. Variability scenarios emphasize '*the differences between individual products*'. In addition, in some cases a simplified version of a family member has to be developed in order to analyze certain situations and behavior, i.e. a prototype.

The document of the commonality analysis is a powerful tool to communicate between key internal stakeholders: marketing department, senior management and development department. This communication is important for the right decisions to be made by the right people for the success of the product family. Ardis et al. (2000) states that since there will be little time to debug every product variant it is essential to have a well-designed architecture and reusable components to develop the family members. The **generic architecture** is of key importance for the success of the product family, since the architecture decides the future possible family members. According to Coplien (1999), in object-oriented system development, design patterns focus largely on variability. This makes design patterns vital for the architecture.

On an economic perspective, the FAST process states that an investment in Domain Engineering is required prior to the initiation of development. However, the developments costs will stay lower compared to when Domain Engineering is not invested in.

3.2.11 Integrated SPPL

Rombach (2005) claims that software processes are still not managed in a systematic way similar to that of SPL engineering, i.e. the effective **reuse** of software artifacts based on proactive organization (of similar artifacts) according to **similarities** and **differences**. Integrated Software Process & Product Lines (SPPL) allows such organization that both artifacts and process to be systematically chosen for a given development project.

As a result of such organization, the processes for a specific project can be tailored according to similarities and differences (similar to application engineering in SPL). Thus, the vision of Integrated SPPL is to be able to choose the needed artifacts and processes based on a set of product and process requirements as well as project constraints. This practice also describes two separate development processes as main characteristics of SPL, namely **Domain** and **Application** Engineering. Software systems are characterized by their **commonalities** and **variabilities**, which are functionalities that are present in **most** systems within that domain and functionalities that are unique to **some** systems within that domain, respectively. The predefined variability choices

(variants) are linked to the corresponding components. From a SPM perspective, this implies that that features are linked to components which will be used for development. In addition, Requirements Engineering within Domain Engineering should focus on defining maximum commonalities and controlled variabilities in order to be address by a stable system architecture on domain level.

3.2.12 Organizational alternatives

As most researchers focus on the technical and process aspects of SPL engineering, Bosch (2001) researched the **organizational alternatives** for organizations employing, or thinking about, a SPL approach. The organizational structure is vital for the **proper execution** of SPL engineering. In extension to the division in Domain and Application Engineering, Bosch (2001) identified and categorized the following organizational alternatives:

- Development department
  This organizational model is focused in one single development departments, i.e. no permanent organizational structure is imposed on the software engineers and architects involved in the SPL. All software staff members can be allocated to do work of any type within the product family. Work that has to be completed is organized in projects that dynamically allocate staff members to certain groups. The project can be for Domain and Application Engineering, both with their goals as developing reusable assets and developing a system, respectively. This organizational model states that both the reusable assets and the finished systems are realized and maintained by one single development department (a single organizational unit). This model is suitable for smaller organizations, i.e. not exceeding 30 software staff members.
- Business units
  This organizational model lays the complete responsibility of developing and the evolution of one or more product from the SPL, in one business unit. The reusable assets needed for development are shared by all business units. The initial developments of these assets are realized through Domain Engineering projects that consist of members from most or all business units. This model optimally ranges the business units between 30 – 100 software staff members. With respect to the evolution of the shared assets, three levels of maturity have been identified (depending on the staff size for each business unit and the amount of shared versus specific functionalities in each system):
  - Unconstrained model
    Any business unit can initiate the extension of the functionalities of any shared asset as long as it adheres to the specifications. The same business unit is responsible for making the new version of the asset available in the assets repository and also for the evolution of the asset.
  - Asset responsible
    An Asset responsible is introduced that verifies the evolution of the asset, based on the best interest of the organization and not that of one single business unit. The Asset responsible is not responsible for new requirements implementation.
  - Mixed responsibility
    Here, each business unit is given the responsibility of one or more shared assets that the business unit makes most (extensive) use of, next to the product already assigned to the business unit. Other business units would need to request their interest whenever an extension is required.
- Domain engineering unit
  This organization model separates the concerns of development and evolution of the shared assets from the development from the end-products. The former is performed by Domain Engineering unit and the latter by the Application (referred to as System or Product) Engineering unit. In addition, this model makes it possible to have one single domain engineering unit (for shared assets) or to have multiple domain engineering units. When it concerns multiple domain engineering units, one unit is responsible for the software architecture and for each architectural

asset (component) a domain engineering unit is assigned that is responsible for the development and evolution for that asset. This model is applicable for organizations where more than 100 software staff members are working on the SPL.

- Hierarchical domain engineering units
  This organizational model creates specialized domain engineering units that are responsible for developing and the evolution of the reusable assets. However, these assets are used as a subset for the product in the SPL, i.e. reusable assets are developed that are necessary for other domain engineering units that will further specify the asset whilst still remaining reusable for specific product line or products. The reusable assets at the top level are often referred to as a platform, providing general and share functionality. This model is applicable for (very) large organizations, software staff numbers in the hundreds, with extensive product families that run long in the future.

3.2.13 Product development projects I

This study investigated project management methods used during the execution of new product development projects (Tatikonda & Rosenthal, 1999). The main problematic challenge was to balance **firmness** and **flexibility** in the project execution phase, in contrast to the project planning phase. First, the influence of project execution methods of *formality, project management (PM) autonomy* and *resource flexibility* on project execution success is researched. Second, the degree of influence of technology novelty on the relationship of project execution methods and project execution success is researched. Project execution success is measured by the degree to which the project achieves its original objectives. For product development projects these objects are technical performance, product unit-cost and time-to-market for development effort. A project is executed properly will most likely have a high level of project execution success. However, the product what it is about can still result in a market failure. One of the suggestions this research gives is that product features might have been chosen incorrectly. This is where product management would claim its responsibility.

The results of the research showed that all the methods, i.e. formality, PM autonomy and resource flexibility positively influence the project execution success. Firmness is achievable through project management formality which makes it possible for general control and review structure for the project. Flexibility is achievable through PM autonomy and resource flexibility which provide a somewhat low-restriction way of working and respond to emerging project uncertainties. This implies that for product development executions to be effective, flexibility within a structure is needed; i.e. having a predetermined structure and allowing enough flexibility within that structure as a way of working. These execution methods work effectively together pointing out that organizations can balance Firmness and Flexibility. As for Technology Novelty, the research results show that Technology Novelty has no significant influence on the relationship between execution methods and execution success. This implies that when organizations are managing a variety of product development projects, broad and similar project execution methods can be used.

As can be consulted above, unfortunately, this research is not focused on product management execution or processes. From product development, this research studied the project execution, i.e. on a project management level. For instance, formality in product development can add effectiveness through providing rules and reviews on the work process with the effect of structure and sequence on the work process. This reduces uncertainties for the project member with respect on what work to do when. This statement is an obvious statement on project management (for new product development) level and we considered it to be out of scope for this research, since we strive to focus on SPM and SPL and the fact that Project Management is too broad topic to be included.

3.2.14 Product Development projects II

This paper describes mainly the planning and execution of two types of projects within product family development. The research investigates (Tatikonda, 1999) project characteristics, development challenges, typical outcomes and success factors. The two types of development projects identified are: **Platform** projects and **Derivative** projects. **Platform projects** are defined as projects that initiate a new product family platform. **Derivative projects** are defined as projects that are extensions to an existing product family platform. The differences between these two project types are explained through two theoretical perspectives, namely *Product/Process Life Cycle Theory* and *Organizational Information Processing Theory.* Product/Process Life Cycle Theory explains that platform projects are more likely to take place early in the product/process life cycle compared to derivative projects that are more likely to occur later in the life cycle. This has implications for instance on technological and market uncertainty and project innovation, all of which are greater during the early phases of the life cycle and decreases later on. Organizational Information Processing theory explains that organizational tasks can be translated to development projects and these tasks differ in the level of unpredictability. This implies that tasks with higher unpredictability require better and more carful (pre-task) planning and these should be executed differently from tasks with lower unpredictability.

The differences between Platform products and Derivative products that have been identified are:

- P1-Project task characteristics: A significance difference is shown in the degree of new technology development for platform projects (higher) versus derivative projects (lower). The data also shows that platform projects have more novel objectives (e.g. performance, costs, time objectives) than derivative projects.
- P2-Market newness: Platform products are perceived as newer by the customers compared to derivative products. Platform products are also intended for markets that are newer to the company and/or industry.
- P3-Project planning: Platform projects have greater commitment from project management in setting project objectives. These projects are expected to be riskier which require more dedication and realistic target setting.
- P4-Project execution: No statistically significant difference was found between the two types of projects when it comes to the approach of project execution. Due to greater levels of unpredictability, platform projects were expected to have different (more organic fashion) project execution approach than derivative projects.
- P5-Project success: No significant difference was found on which project is more successful. Derivative projects were expected to be more successful due to presumed lower technology novelty, and project complexity.
- P6-Project smoothness: The results show no significant difference between the two project types when it comes to the smoothness of the project execution. Since platform projects have greater risks and in turn greater unpredictability, it was posited that platform projects would have lower project execution smoothness.

A result from this research points out that a single product development process can be employed for both platform projects as well as derivative projects. However, modest customization of the development process is needed for the project type, i.e. customize the process as needed for the corresponding project type which (from this research) result to be relatively little. In practice, both platform and derivative projects are generally managed in the same way.

As the previous one, this study has a project management perspective on product development, namely based on Platform and Derivative projects. In addition, main aspects that is of importance such as project complexity, market newness, project risk, formality, project evaluation of personnel, engineering tools and trainings, etc. points out that the research is focused on project management

(albeit for product development) rather than product management. The identified development situations (Platform and Derivative) are relevant.
However, not covering details such as product feature/ or functionalities collecting or planning of product development with respect to functionalities, implies a lack of depth into product management processes, what we are searching for. Instead, this research covers these development situations from a project management point of view. Like the previous study, this is out-of-scope for this research.

3.3 Summary

In this section (chapter 3) the theoretical foundation for this research has been set. The most essential topics have been discussed in detail:

- Software Product Management, as defined by the SPM Competence Model.
- Software Product Lines, as researched by numerous authors.
- Product development situations, as what was found most relating to this research.

Below, table 8 gives overview of the useful literature, categorized in SPL-Literature and the authors who recognized the practice as part of SPL engineering.
NOTE: not all consulted literature in the literature base was useful for this research, in other words within our scope of SPM and SPL. Most of the used SPL-literature refers to Domain and Application Engineering. This is one of the practices most recognized in SPL engineering next to Variability management and Architecture.
Despite for being out of scope for this research, the reference (generic) architecture of a product family is unmistakable, i.e. being technical on a development level is not considered within the scope of this research. However, literature points out that it is of key essence for the success of a SPL. It is for this reason we include Architecture in the mapping of the SPL-literature and investigated how and which part of literature we could map to SPM; our main focus. This is explained in the next section.

Table 9: SPL-literature overview with corresponding references.

SPL-Literature	Reference
Domain Engineering & Application Engineering	Pohl, Böckle, Linden(2005)
Core asset development & Product development and Management	Clements (1999, 2001); Northrop (2002)
Variability Management	Bosch (2000), Svahnberg (2000), Gurp (2001), Jaring (2002), Halmans (2003), Czarnecki (2004)
Requiline	Maβen & Lichter (2004), Taborda (2004)
Generalized release planning for SPL	Taborda (2004),
Product line Portfolio Planning using Quality Function Deployment (QFD-PPP)	Helferich, Herzwurm, Schockert (2005)
Requirements taxonomy	Schmid & Eichelberger (2007)
PuLSE	Bayer, Flege, Knauber, Laqua, Muthig, Schmid, Widem, DeBaud (1999)
Product Derivation Framework	Deelstra, Sinnema, Bosch (2003, 2004)
Integrated Software Product & Process Line (SPPL)	Rombach (2005)
Family-oriented, Abstraction, Specification and Translation (FAST) process	Ardis, Daley, Hoffman, Siy, Weiss (2000)
Business, Architecture, Process and Organization (BAPO)	Linden (2002)

Organizational alternatives	Bosch (2001)
Product Development projects I	Tatikonda (1999)
Product Development projects II	Tatikonda & Rosenthal (2000)

4 SPL-Capability process

The SPL-literature presented in the previous section is represented by, what we call, SPL-practices. These SPL-practices all have their own definitions, methods, activities, processes and deliverables for SPL development that have relations to product management which are extracted in order participate in the mapping process, which we call SPL-Activities. Figure 7 illustrates the relationship of SPL-literature, SPL-Practice and SPL-Activity. The latter two are elaborated later on.

Figure 6: n-to-n relation between SPL-Literature, SPL-Practice and SPL-Activity.

In the mapping process, comparisons will be made between the literature findings and the SPM Competence Model. Depending on the degree of commonness, the knowledge of the Competence model can be applied accordingly to the SPL-Practice. Depending on the degree of differences, candidates for modifications or improvements can be identified that need to be included in the Competence model. Since academic research specifically on product management for SPL is limited, the mapping with the SPM Competence model is beneficial. The benefit is that the knowledge and practice of a business function, focus area or capability applies for a SPL-practice (or activity) when the two are similar or even equal. Thus, the mapping implies that the link the SPL-Activity has towards the specific part of the SPM Competence Model is registered.

To be exact, the actual mapping will occur to the SPM maturity matrix. As mentioned previously, the SPM maturity matrix is the detailed representation (including capabilities) of the SPM Competence Mode in the form of a matrix. In order to compare, analyze and properly perform the mapping, SPL-practices are described on similar level as the SPM Maturity Matrix. During the mapping, a commonality implies no modifications, whereas a difference implies possible candidate improvement to the maturity matrix.

4.1 Literature analysis

Once the SPL-literature was selected and studied, it became clear that most literature were (too) development-focused. At first, these were focused on the actual development processes (technical processes with the purpose to develop the end-product) of SPL and not directly related to SPM. This finding led to the categorization of the literature base into: *Development* and *Product Management*. Product management literature was identified by the focus on processes similar to the SPM Competence Model or processes that surrounded, and was clear input for, development. The categorization resulted as follows [amount of papers]:

- Development [19]
- Product Management [10]
- Development & Product management [15]
- Development situations [11]

4.1.1 SPL-Practice identification

This categorization revealed that the literature related to product management had the least amount of papers. This was the category that was firstly used to identify the SPL-practices that would be relevant for this research. A SPL-Practice describes the customary or habitually actions or process of a particular topic regarding SPL. This resulted in 5 distinct SPL-practices [papers on topic]:

- Product line Release planning [4]
- BAPO Model [2]

- Product Portfolio Planning – QFD [2]
- Product Derivation [3]
- Architectural design [6]

The SPL-practices were named after the main content process, a central model or framework. The SPL-practice 'Architectural design' has the exception that it is not related to SPM. However, in the majority of the product management literature this process was repeatedly mentioned as of vital importance for the development and success of a product line. Hence, its inclusion. Below we give an example of a Development and Product management category and their rationale:

Table 10: Example of SPL-literature categorization into Development and Product management

SPL-Practice	Development	Product Management	Rationale
Product Derivation		X	The process consists of two phases: the initial and the iteration phase, which are managed by Requirements Engineering throughout the entire process of derivation. Based on requirement input the initial phase has the responsibility to create the best fit configuration. In the iteration phase, the configuration is iterated a number of times until the end-product adhere to the requirements set up-front by Requirements Engineering.
PuLSE	X		PuLSE exists of three main elements: Deployments phases, Technical and Support Components. Deployment phases describe the logical stages and activities performed to set up and use the product line, e.g. scoping and design for the architecture and development infrastructure. Technical Components provides the technical know-how that is needed to realize and operationalize the product line development. The Support Components are the bundles of information or guidelines which provides a better adaption, deployment and evolution of the SPL.

The amount of identified practices by taking the 'Product management' category was not satisfactory; practices discussed familiar concepts such as product feature identification or product planning, however details were not always clear. This was the reason to also take an in-depth look at the Development category. The literature gathering process assured that another search would not result in more new SPM-related literature. However, when a particular topic covered both SPL and SPM-practices and described possible improvements, this topic was re-searched with the intention to find more literature on that particular topic. Most of the time, this re-search did not result in more 'new' literature. Hence, the Development-related papers were analyzed to filter out the SPM-related practices. Development processes, models, frameworks, design techniques as described in literature were studied and the SPM essence of each part was analyzed for relation to the SPM Competence Model. This resulted into 7 more SPL-practices [papers on topic]:

- Product line Release planning [4]
- BAPO Model [2]
- Product Portfolio Planning – QFD [2]
- Product Derivation [3]
- Architectural design [6]
- *Domain & Application Engineering [21]*
- *Core asset, Product development & Management [7]*
- *Variability management [17]*
- *Requirements practices [10]*
- *PuLSE [2]*
- *RequiLine [1]*

- *Requirements taxonomy [1]*

When taking both Development and Product management categories it became clear that there are certain SPL-practices that great majority of the literature relates to. These were two practices, namely, Domain and Application Engineering and Variability management. In addition, most SPL-practices are mentioned or discussed in multiple papers (figure 7). It can be noted that the last two practices, RequiLine and Requirements taxonomy, only has 1 paper each. Nevertheless, these practices have a direct relation to SPM that are candidate improvements.

4.1.2 SPL-Activity identification

The 12 identified SPL-practices were candidate practices that could provide knowledge to be included in the SPM Maturity Matrix. During the analysis of the literature, each SPL-practice was further described by its collection of activities (or method or processes description) that forms the building blocks of that practice, its characteristics. We refer to these as SPL-activities.
The reasoning behind this naming is that when a SPL-practice is broken down into details and its core essence is analyzed, we looked for the capacity for being useful for a specific purpose (SPL management) that can be expressed in an action, step or instruction, i.e. activity. A SPL-Activity is described as similar as possible as the SPL-Practice describes it, i.e. using the same keywords and terminology. Whilst SPL-activities and SPM-capabilities are described on similar level in order to create a fair comparison, SPL-activities are not called SPL-capabilities on purpose. The naming of 'SPL-Capability' is reserved for a later step in the mapping process.

Every SPL-practice has a bundle of SPL-activities, i.e. one or more activities. However, some SPL-activities are also part of other SPL-practices, i.e. the activities are not unique for the SPL-Practice. This creates the *n-to-n* relationship which is illustrated in figure 7. The naming of these activities is kept relatively short and usually can be related to The SPM Competence Model. Below, in table 11, we present the identified SPL-activities with the corresponding SPL-Practice.

For a clear overview, the amount of SPL-activities is shortened in table 11. SPL-practice with more than five identified SPL-activities is only presented with the five SPL-activities that are most relevant based on SPL and SPM importance of that practice. The complete table with all the identified SPL-activities is represented in Appendix A. This will not be at the expense of understandability, since only less data is presented in one table and the whole table is included in the appendix.

Table 11: Identified SPL-practices and SPL-activities.

SPL-practice	SPL-Activity
Domain Engineering (DE) & Application Engineering (AE)	Creation of roadmap for (common and variable) product features
	PM deals directly and firstly with Requirements engineering (RE) (first on domain level afterwards on product or application level)
	RE differentiate between common and variable features
	Variability management is dealt with in RE
	Architecture design is driven by RE
Core asset, Product development and Management	(Technical) Management monitors the processes of Core asset (DE) and product development (AE)
Variability management	Variability in SPL is determined by the variable features between the SPL-members
	Requirements traceability is needed
	Reusable Components represent a set of functionalities of the products
	A component in the architecture implements a coherent domain or set of functionalities.
	Identifying variability is often based on analyzing commonalities and differences between SPL-members
RequiLine	RM differentiate between common and variable requirements
	Requirements/feature traceability is needed

Generalized Release planning for SPL	Requirements allocation and traceability
	Release planning for components
PPP-QFD	Requirements identification AND prioritization by customers
	RM identifies Product line members
PuLSE	Management path SPL future
	Product map defines product line scope
	Architecture is driven by requirements
	Requirements are also used for product validation
	Design and coding is validated against the architecture
Product Derivation Framework	Requirements traceability is essential
	Product configuration is validated against the requirements
	Product architecture is derived from the reference architecture
	Product roadmapping
Integrated SPPL	SPL engineering exists of 2 separate development processes: DE and AE
	SPL engineering promotes proactive reuse of pre-designed commonalities and controlled variabilities within a family of systems
	Commonalities and variabilities are implemented through a components architecture
SPL FAST Process	The FAST process exists of two phases: DE and AE
	During commonality analysis example scenarios are used to explore differences between SPL-members
	Prototyping a SPL-member makes it possible explore differences between the members
BAPO	RM is input for the architecture design
	Traceability of requirements is vital in RM
	Business: Identity
	Architecture: Reuse levels
	Architecture: Product quality
Requirements taxonomy	Requirements are categorized in: Commonality, Variability and Product-specific

4.2 The mapping process

The total amount of SPL-activities identified is 84 from the 12 identified SPL-practices (see appendix A). However, this does not mean all SPL-activities are unique, since various SPL-activities are identified in more than one SPL-Practice. This implies that some SPL-activities are redundant; 15 SPL-activities in total. However, this does not have a negative effect since a redundant SPL-Activity only adds weights of importance to the activity for being identified by multiple SPL-practices.

The purpose of the mapping process from business function-to-focus area-to-capability is to know exactly to which part of the Maturity Matrix the SPL-activity links to. We want to know this to know on which level the knowledge improvements can be applied. The following steps are performed during the mapping process:

1. Comparison with the SPM Maturity Matrix.
   A SPL-Activity is compared from a generic level to more a specific level in the Competence Model, i.e. the activity is compared to the business functions to check where the activity best fit according to the definition of the business functions in (Bekkers & Weerd, 2010). Next, the activity is compared to the focus areas of that business function to check where it best fit, also based on the definition of the focus areas in (Bekkers & Weerd, 2010). Finally, the activity is compared to the SPM-capabilities of that particular focus area to check which one the activity resembles most, also according to the description of the capabilities from (Bekkers & Weerd, 2012). The aim is to compare the activities on an as-most-specific level as possible, i.e. capabilities level. However, if this is not possible, then the most specific comparison possible is made, e.g. if SPL-Activity x is compared to the focus area of Requirement Identification and it cannot be related to not one capability, the linking stays at Requirements Identification. This comparison is registered for every activity of each SPL-Practice.

2. A status is defined.
   A status describes the link-type between the SPL-activity and the SPM Maturity matrix. This can be a Similarity, a Candidate improvement or it can be Neutral. In order to remain clear and understandable, colors are used for the defining of the statuses. See table 12 for elaboration.
3. A rationale is defined.
   A rationale describes the reasoning behind the decision made on the status regarding the link of the activity and the Competence Model.
4. Architecture or Organizational
   As last step, SPL-activities that are related to Architectural design for SPL or related to changes, processes or structure on an organization level regarding SPL, are registered. These two aspects, especially Architecture, have been noticed to be essential for the overall success of the SPL.

Table 12: Statuses explained that a SPL-activity can obtain and eventually an Architecture or Organizational focus.

Status	Color	Criteria
Similarity	Green	The description of an SPL-activity is similar or equal to the description of the business function, focus area or capability comparing keywords and terminology. Verbs such as gather, identify, allocate, planning, etc. and SPM terms such as requirements, release, validation, roadmap, etc. are compared.
Candidate improvement	Blue	The description of the activity is SPM-related (keywords and terminology), however it is not described by or incorporated in the Competence Model or only partially, with respect to SPL. Verbs such as gather, identify, allocate, product planning, etc. and SPM terms such as requirements, release, build, validation, roadmap, etc. are compared.
Neutral	Grey	Not a convincing Similarity nor a convincing Candidate improvement, an in-between
SPL-Architecture	A	SPL-Activity that describes input, design process or characteristics of the product line architecture.
Organizational	O	SPL-activity that describe processes essential for SPL-engineering on an organizational level.

After the above mentioned steps 1 through 4 has been completed for a SPL-Activity, a mapping is created for that activity. Below, we present the SPL-activities presented in table 13 and their mapping as described above. Note: same as table 11, this table (table 13) has the same limitation on SPL-activities. However, the same activities are presented as in table 11 in order to clearly present the mapping process. For a complete overview of the mapping, please see Appendix A.

Table 13: SPL-activities mapping to the SPM Maturity Matrix.

SPL-practice	SPL-Activity	SPMCM-mapping	Status	Rationale
Domain Engineering (DE) & Application Engineering (AE)	Creation of roadmap for (common and variable) product features	Product planning : PR		Improvement: creation of product roadmap based on common and variable product features of the intended SPL-members. No details are given on the timespan of the roadmap except for "as far as foreseeable". This Roadmap creation is similar with the focus area Product Planning, except for the product features and timespan information.
	PM deals directly and firstly with Requirements engineering (RE) (first on domain level afterwards on product or application level)	Requirements management	O	Neutral: mostly stating Requirements Gathering and Organizing aspects and describe the process on an organizational level. PM defines the common and variable features of the SPL and the members and includes these in the roadmap, which serves as the scope for DRE. Afterwards, PM defines which products should be derived in ARE.
	RE differentiate between common and variable features	Requirements management: RG		Improvement: RM should differentiate between common and variable requirements when identifying product features. Identify more common than variable requirements, as variable requirements assure more complexity. However variable requirements are necessary for the essence of the SPL, which is the variability each product or application will have.
	Variability management is dealt with in RE	Requirements management: RO		Improvement: RM should manage the variability, which is identifying, documenting and modeling the variable requirements. RM explicitly document and model variability (external variability= visible to customers, possible to choose variants). This entails variable requirements and modeling (variation points, variants and their relationships) of this variability. Here the variability diagram is created and presents the differences between the members.
	Architecture design is driven by RE	Requirements Management	A	Improvement: RM should implement some practice towards architectural design. For instance, quality requirements should count for the architectural design or grouping of requirements according to architectural concerns. Common and variable requirements and the variability model are passed onto Domain Design which translates the requirements to technical solutions in the SPL architecture. Especially quality requirements (performance, security, usability, etc.) are the drivers for architectural design. The Variation/variability in requirements often results in variation/variability in the architecture. Component frameworks are used to support the various types of quality requirements, i.e. frameworks are used to model SPL requirements in a structured manner into components with their relationships. It also incorporates, properly, quality requirements such as flexibility, maintainability, evolvability.
Core asset, Product development and Management	(Technical) Management monitors the processes of Core asset (DE) and product development (AE)	Overall	O	Improvement: Get (top) management more involved in the DE & AE processes, e.g. by monthly reports or management involved, e.g. as a stakeholder for instance at the end of each Sprint when following agile Scrum. However, it is relevant for the overall success of

				SPM processes.
Variability management	Variability in SPL is determined by the variable features between the SPL-members	Requirements management: RI, RO		Improvement: Determine which requirements will differentiate between the SPL-members. Focus area's RI for identifying the variation in the requirements, RO for organizing the common and variable features and the modeling there of. The identified variability has to be modeled by a variability-modeling technique, e.g. Feature Modeling technique.
	Requirements traceability is needed	Requirements management: RO:B		Improvement: Register in which (core) asset or component a requirement will be implemented as extra requirement data. The capability of RO:B logs requirements' data expect for in which asset a requirements will be implemented.
	Reusable Components represent a set of functionalities of the products	Requirements management		Improvement: Requirements can be organized according to components explicitly. This statement states a set of functionalities or requirements form a component.
	A component in the architecture implements a coherent domain or set of functionalities.	Requirements management: RO	A	Neutral: A component consists of functionalities that are closely related or together form a solution. Similar to the Requirements organizing focus area.
	Identifying variability is often based on analyzing commonalities and differences between SPL-members	Requirements management: RI, RO		Improvement: Commonalities and differences identification and analysis should be part of RM, since it comes down to common or different features and requirements
RequiLine	RM differentiate between common and variable requirements	Requirements management: RI		Improvement: RM should differentiate between requirements for all SPL-members (common) and requirements for specific SPL-member (variable)
	Requirements/feature traceability is needed	Requirements management: RO:B		Improvement: Features should be linked to the SPL-member they are implemented in. This makes it possible to trace requirements down (history of requirements implementation) and have the knowledge of how an issue has been solved before.
Generalized Release planning for SPL	Requirements allocation and traceability	Requirements management: RO:A, B		Improvement: Bundle requirements that fit together and register where this bundle will be implemented, organized per SPL-member inclusion. Both components and end-product are linked in the Release matrix. This implies requirements being allocated to components and finally the end-products. Traceability too, requirements can easily be traced in which product they were implemented
	Release planning for components	Release planning: RD:B, C		Improvement: Plan the release of components and products in the Release matrix, as is needed to meet the release date (the component producer is the one responsible for requirements prioritization).
PPP-QFD	Requirements identification AND prioritization by customers	Requirements management: RG:E – Release planning: RP:C		Similarity: Requirements for the SPL are firstly gathered from existing and potential customers. These requirements are analyzed and sorted. Secondly, the existing and potential customers are asked to prioritize the requirements.
	RM identifies Product line members	Requirements management: RO – Release planning: RP		Improvement: Requirements can be sorted and organized in such a way, that segments can be extracted. Based on the prioritized requirements, customer segments are derived using cluster analysis. Each product line member is identified using the rule 'one product line member per customer segment'. Experts provides input on a technical level.
PuLSE	Management path SPL future	Portfolio management		Neutral: The product line scope is initiated by business objectives defined by the stakeholders. Management defines the business objectives that drive the SPL initiation.

	Product map defines product line scope	Product planning: RI		Improvement: to present requirements with SPL-members together with other valuable information for the product line, i.e. costs, benefits, market, objectives and competitors in a product map. A matrix with the SPL-members in the columns and the characteristics in the rows together with information on the market, costs, competitors and benefits.
	Architecture is driven by requirements	Requirements management	A	Improvement: common, variable and product-specific requirements are used to initiate the architectural design of the SPL, i.e. concepts are determined and modeled according to their relation and product line scope. The aim is to define a domain-specific SW architecture that covers the existing products and future SPL-members
	Requirements are also used for product validation	Release planning: RBV		Similarity: validation of the end product is performed to assure product quality according to the requirements set beforehand.
	Design and coding is validated against the architecture	Release planning		Improvement: the reference architecture is used for the validation of the design models and SW-coding to validate if the limitations and structure of the architecture are met.
Product Derivation Framework	Requirements traceability is essential	Requirements management: RO:A, B		Improvement: requirements are also linked to the core asset in which they've been implemented. Requirements are organized based on shared core assets is a similarity.
	Product configuration is validated against the requirements	Release planning: RBV		Similarity: validation of the end-product, to assure product quality, according to the requirements set beforehand.
	Product architecture is derived from the reference architecture	Architecture	A	Neutral: The product architecture is derived from the product line reference architecture, for as much as possible.
	Product roadmapping	Product planning: RI, PR		Similarity: the domain and scope of the SPL as well as its future developments (evolution) are predicted in combination with a technology scope in a roadmap. However, no timespan is given.
Integrated SPPL	SPL engineering exists of 2 separate development processes: DE and AE	-	O	Neutral: One process, DE, is responsible for developing reusable components and set up the product line development platform. The other process, AE, uses mainly the reusable components developed by DE to build the end-products, and tailors where this is needed.
	SPL engineering promotes proactive reuse of pre-designed commonalities and controlled variabilities within a family of systems	Requirements management: RG		Improvement: pre-designed commonalities and controlled variabilities are common and variable features between the SPL-members. These should be handled in RM
	Commonalities and variabilities are implemented through a components architecture	Requirements management	A	Neutral: common and variable features are organized into components. The architecture describes how components should be implemented and how they relate to each other.
SPL FAST Process	The FAST process exists of two phases: DE and AE	-	O	Neutral: DE and AE are recognized as the two main processes essential for SPL engineering. Similar to other SPL-practices.
	During commonality analysis example scenarios are used to explore differences between SPL-members	Requirements management: RI		Improvement: use these example scenarios (techniques) in order to further analyze common and variable features when this is needed for certain products.

	Prototyping a SPL-member makes it possible explore differences between the members	Release planning		Improvement: Prototyping allows for deeper identification and analysis of variable features and other aspects between SPL-member when scenarios are not sufficient.
BAPO	RM is input for the architecture design	Requirements management		Improvement: The design for the reference architecture receives input from RM, i.e. the functional and quality requirements form the design of the reference architecture. Both commonalities & variability (variation points and variants) should be modeled in the architecture. The reference architecture defines the components (mandatory, optional, and alternative), component interrelationships, constraints, and guidelines for use and evolution in building systems of the SPL.
	Traceability of requirements is vital in RM	Requirements management: RO:B		Improvement: RM should trace requirements to know in which assets they are implemented, for maintenance reasons and a complete manageable process. Traceability is connected with configuration and version management for the configurations and version of the particular assets and components.
	Business: Identity	Product planning: CAR	O	Neutral: Existing family assets are reused in product development for opportunistic reasons. Likewise, make/buy/mine/commission SPL assets are only done for opportunistic reasons. This is information that is not to be missed. However, it is not more than that.
	Architecture: Reuse levels	Product planning	A	Neutral: asset sharing is only beneficial when the commonalities are clear to be exploited. Domain-specific components can be acquired from external sources if this is more beneficial (less effort) than building them.
	Architecture: Product quality	-	O	Neutral: intra-organizational reuse of assets takes place through a platform which provides domain functionality that is applicable for all products, i.e. commonality. Non-commonalities are implemented in individual application or product (product derivation)

4.3 SPL-Capabilities

In the previous section we presented how and where the identified SPL-activities map to in the SPM Maturity Matrix. We now know which SPL-activities we can make further use of in order to incorporate the SPL-knowledge in the Maturity Matrix. From the three types of statuses presented in table 12, Candidate improvement is most interested for us. This mapping's purpose is to identify SPL-knowledge that is not provided by the SPM Maturity Matrix, whilst being applicable to do so. Most SPL-activities are described on the same detail-level as SPM-capabilities, i.e. describing an action that has to take place with a specific goal. Usually this goal is represented by the focus area where the capability is grouped in together with other related capabilities. SPL-activities have been described in this way in order to compare them with SPM-capabilities. In addition, when a SPL-Activity is mapped as a Candidate improvement it can thus be rewritten in the same format as the SPM-capabilities or as a focus area or a business function. By doing this, we can incorporate new knowledge (SPL) into the SPM Maturity Matrix. The Maturity Matrix would then have extra SPL-specific capabilities, focus areas or business functions.

The Similarity-mapping has little further implications. When a SPL-Activity is similar or equal to the business function, focus area or SPM-Capability, i.e. no significant differences are identified in the descriptions comparison, then it can be stated that the particular SPM-business function, focus area or capability is applicable for SPL-engineering also. The Neutral-mapping can have different implications. As explained in table 12, when a SPL-activity is neither a convincing Candidate improvement nor a convincing Similarity, it is automatically a Neutral. When a SPL-Activity is mapped as a Candidate improvement or Neutral, the SPL-Activity can also be linked to SPL-Architecture or Organizational. These SPL-activities are more focused on the architectural or organizational importance of SPL-engineering. Nevertheless, SPL-literature assured that these two topics were as essential as the product management ones, especially SPL-Architecture.

Below we present a summary of the mapping process: a summary of all the SPL-activities with their mapping in the Maturity Matrix. In table 14 we present the same amount of SPL-activities we have used in the previous section, i.e. table 11. The complete mapping process is included in Appendix A. Note that the mapping to the Maturity Matrix is presented in abbreviations in the form of: business function:focus area:capability. For instance, Product planning:Product roadmapping would be PP:PR, Requirements management:Requirements organizing:capability B would be RM:RO:B.

Table 14: Summary of the SPL-activities mapping process.

SPL-practice	SPL-Activity	Similarity	Candidate Imp.	Neutral
Domain Engineering (DE) & Application Engineering (AE)	Creation of roadmap for (common and variable) product features		PP:PR	
	O: PM deals directly and firstly with Requirements engineering (RE) (first on domain level afterwards on product or application level)			RM
	RE differentiate between common and variable features		RM:RG	
	Variability management is dealt with in RE		RM:RO	
	A: Architecture design is driven by RE		RM	
Core asset, Product development and Management	O: (Technical) Management monitors the processes of Core asset (DE) and product development (AE)			Overall
Variability management	Variability in SPL is determined by the variable features between the SPL-members		RM:RI, RO	
	Requirements traceability is needed		RM:RO:B	
	Reusable Components represent a set of functionalities of the products		RM	

	A: A component in the architecture implements a coherent domain or set of functionalities.			RM:RO
	Identifying variability is often based on analyzing commonalities and differences between SPL-members		RM:RI, RO	
RequiLine	RM differentiate between common and variable requirements		RM:RI	
	Requirements/feature traceability is needed		RM:RO:B	
Generalized Release planning for SPL	Requirements allocation and traceability		RM:RO:A, B	
	Release planning for components		RP:RD:B, C	
PPP-QFD	Requirements identification & prioritization by customers	RM:RG:E – RP:RP:C		
	RM identifies Product line members		RM:RO – RP:RP	
PuLSE	Management path SPL future			-
	Product map defines product line scope		PP:RI	
	A: Architecture is driven by requirements		RM	
	Requirements are also used for product validation	RP:RBV		
	Design and coding is validated against the architecture		RP	
Product Derivation Framework	Requirements traceability is essential		RM:RO:A, B	
	Product configuration is validated against the requirements	RP: RBV		
	A: Product architecture is derived from the reference architecture			
	Product roadmapping	PP:RI, PR		
Integrated SPPL	SPL engineering exists of 2 separate development processes: DE and AE			
	SPL engineering promotes proactive reuse of pre-designed commonalities and controlled variabilities within a family of systems		RM:RG	
	A: Commonalities and variabilities are implemented through a components architecture			RM
SPL FAST Process	O: The FAST process exists of two phases: DE and AE			
	During commonality analysis example scenarios are used to explore differences between SPL-members		RM:RI	
	Prototyping a SPL-member makes it possible explore differences between the members		RP	
BAPO	RM is input for the architecture design		RM	
	Traceability of requirements is vital in RM		RM:RO:B	
	O: Business: Identity			PP:CAR
	A: Architecture: Reuse levels			PP
	O: Architecture: Product quality			

As can be seen in table 14, the column of Candidate improvement is highlighted. These SPL-activities will contribute in the coding into SPL-capabilities. The process of transforming a SPL-Activity into a SPL-capability is rather simple: a SPL-Activity that is mapped as candidate improvement will be coded in the same format as a SPM-capability. After this coding, the SPL-Activity becomes a *SPL-Capability*. A SPM-Capability is coded according to the following attributes (Bekkers et al., 2010a):

- Title

- Goal
  The goal to be achieved by possessing the capability
- Action
  The action required by the organization in order to perform the capability.
- Prerequisite(s)
  Capabilities that need to be achieved before the capability in question can be achieved
- References
  Related literature supporting the organization in the implementation and understanding of the capability.

We coded the SPL-activities into SPL-capabilities and categorized them hierarchical as this is done in in the Maturity Matrix, i.e. Business functions having Focus areas having Capabilities. As can be noted in the mapping process, SPL-activities are not unique, in other words, various SPL-practices identified similar and even equal SPL-activities. When taking the SPL-Capability coding into consideration, this implies that multiple activities can contribute to the coding of one SPL-Capability. In Appendix A, where the full mapping can be seen, each activity has a unique identification number; an ID. The activities that contribute to each SPL-Capability are noted in brackets behind the title and can be traced back in the complete mapping table (Appendix A).

The fields of prerequisites and references are filled in a later section, since the SPL-capabilities will be evaluated by experts before actually being implemented in the Maturity Matrix. Below we present the Candidate improvements SPL-Activities that are coded into SPL-capabilities. Note, not all the SPL-capabilities are presented, since this will result in a too large amount of (more) tables. However, Appendix B presents all the SPL-capabilities. For each business function a selection of SPL-capabilities has been made based on SPL-essence to present. The SPL-Activities regarding SPL-Architecture and Organizational aspects have also been coded as SPL-Capabilities. At this point of the research these are only recognized as SPL-Capabilities, however they do not yet belong to a specific business function. The reasoning is to get the SPL-capabilities through expert evaluation and to process the feedback and have more knowledge to place the Architecture and Organizational SPL-capabilities properly.

Requirements management

Requirements gathering

Title	Basic product line scoping (7, 8, 10, 47, 48, 54, 59, 64)
Goal	Define product line features to support the scope
Action	Requirements management defines *Common*, *Variable* and *Product-specific* product features. A Common feature is present in all or most products. A Variable feature is present in some products only. A product-specific feature is present in only one individual product (customer wish). A product feature is a logical unit of behavior that is specified by a set of functional and quality requirements, implying a feature is defined by multiple requirements. By defining the different features, and thus requirements, and in which product they will be present (variable features), the different product line-members can be identified. The aim is to define more common than variable requirements, as variable requirements assure more complexity. However variable requirements are necessary for the overall variability of the SPL.

Requirements organizing

Title	Product line features organizing (22,23,24)
Goal	Organize features according to (reusable) components
Action	Organize features together that serve the same purpose or functionality to form components, i.e. features that complete a function of a component are grouped together. A (Reusable) component represents a set of closely related functionalities/features that form a product solution.

Title	Variable feature management (13,18,41,42,43,44,45)
Goal	Manage variable product line features properly.
Action	Variability (variable features) is identified and explicitly modeled and documented. This is referred to as Variability management. Identifying variability is often based on analyzing commonalities and differences between SPL-members; especially external variability which is visible to end-users and possible for them to choose variants. This happens while gathering, identifying and defining product features. The variable features are destined to be implemented in (only) some individual SPL-members, unlike the common features, which are implemented in all members. The variable features and the variances are modeled into a Variability diagram with Variation points (decision points), variants (a decision) and the relation thereof with proper textual documentation. This is also known as Feature Modeling technique. The purpose is to have a clear view of the variability of the product line and to be able to manage it, since this is vital for the product line success.

Release planning

Requirements prioritization

Title	Product line features prioritization (58)
Goal	Product line features prioritization
Action	Prioritize the features that will be implemented in end-product from the next release on by assigning priority to them (prioritization techniques can be used). This prioritization is performed with the end-product(s) in focus and which component is required to complete the product. This implies that features would be implemented in components and these components would complete the end-product. Prioritization is necessary, since not all features can be implemented, due to costs, resources and market introduction deadlines. Hence, the features with the desired priority will be included in the particular components.

Release definition

Title	Product line release definition (4,58)
Goal	A selection of features for implementation based on priority
Action	A practical selection of the features is made given the limitations on engineering resources, based on the priority assigned. The function and essence of the components is also considered when making the selection. The selection is defined textually which will be necessary for further steps.

Release build validation

Title	Architectural release validation (71)
Goal	Release validation by architecture – Release quality assurance
Action	The design and coding of the SPL-members (the build) is validated through the product line architecture before the actual release is launched. The design and software code have to adhere to the limitations and structure of the product line architecture. This validation is performed by the department(s) who is (are) responsible for developing and maintaining the product line's architecture(s). The product line's architecture is vital for achieving the business goals set up front, when management decides to engage in a software product line development approach. Hence, the necessity for validation through architecture.

Product planning

Product roadmapping

Title	Product line roadmapping (3,29,46,105)
Goal	Define the scope of the software product line through a roadmap.
Action	A roadmap is created detailing the anticipated products of the product line, its members, as far as foreseeable. The members are represented by the components (if possible) of which they are built off, i.e. multiple features form a component, whereas a feature abstract from requirements. The product line features, common and variable, should be predicted for a time span of 5 years. However, other authors (e.g. Svanhberg et al.) believe this is not practical, since a great amount of

	the future requirements (not technology shifts and/or other development changes) of the product line cannot be predicted.

Core asset roadmapping

Title	Core asset usage (3,29,46,105)
Goal	Intra-organizational reuse of core assets – exploiting core assets.
Action	Increasing commonalities will need to be managed in order to be exploited properly. The managed commonalities are developed into fundamental components (core assets). These core assets, which mostly contain domain functionality, amongst other shared assets, are reused by other (internal) departments through the product line platform. Features that are shared by sufficient members are included in the core assets, whereas features shared by only few members are developed as part as product derivation. This ensures knowledge sharing and more efficient product development as more reuse is taking place. This is a typical of product lines practice.

Portfolio management

Product lifecycle management

Title	Financial Product line scoping (106)
Goal	Scope the product line with information on costs and profits.
Action	The decision whether a product will or will not be part the product line scope is based on the expectations of the ROI. If these are beneficial for the organization, the product will be part of the line, otherwise it will be declined. This can also be applied to features, i.e. features are dropped according to their expected added value or revenue.

Not coupled to any business functions

Product line architecture

Title	Reference architecture construction(14,24,27)
Goal	Create product line reference architecture.
Action	Create the product line reference architecture. Role of the architecture is to describe the commonalities and variabilities of the products in the product line and to provide the overall structure. Common and variable features, represented by the components, are the main drivers for architectural design together with quality requirements such as performance, security, usability, etc. These are also represented in the architecture. Component frameworks are used to support the various types of quality requirements, i.e. frameworks are used to model features in a structured manner into components with their relationships. The reference architecture has to solve issues of variability and reusability. In addition it also properly incorporates quality requirements such as flexibility, maintainability, evolvability. When common and variable features are considered in a very early stage then more flexibility is assured for the product line. As features are represented in components, the components in the architecture implement a particular/coherent domain of functionality, e.g. the network communication domain.

Organizational

Title	Domain & Application engineering (6,80,87)
Goal	Create a product line environment
Action	Create two processes: one process, Domain Engineering, that is focused on developing reusable artifacts and a product line environment (textual and modeled requirements specification, architecture, design, variability model, software components, tests plans, and more) which forms the development platform; and one process, Application Engineering, that focus on developing sellable end-products that are built from the artifacts developed in Domain Engineering

Title	Requirements engineering planning (5,7)
Goal	Plan requirements engineering for product line developments.

Action	Product management controls Requirements engineering directly, first on domain level afterwards on product (or application) level. Product Management defines the common and variable features of the product line in the roadmap, which serves as the scope for further requirements engineering tasks such as detail specification and modeling. Afterwards, Product Management defines which products or applications should be derived in Application Requirements Engineering, implying which requirements will be implemented for which product, individually (variable features)

5 Expert evaluation

In this chapter, we present the evaluation of the SPL-capabilities that we discussed in the previous one. In the Research approach chapter we mentioned that the identified SPL-practices which finally provided the SPL-capabilities would need to be evaluated before being implemented in the SPM Maturity Matrix. First, we reflect on the scope of the evaluation. Second, we present the questionnaire we used and its contents. Finally, we discuss the evaluation results and present the modifications to the SPL-Capabilities.

5.1 Evaluation scope

The total SPL-capabilities that were defined through the mapping process resulted in 21 capabilities. In order to make a SPL-Capability as comprehendible as possible for the evaluation it included some textual explanation. This was included in the Action-attribute of the capability, even though the Action-attribute is supposed to describe only the action required to achieve that particular capability. The consequence was that each SPL-Capability was larger in text than was actually needed, in other words some text was more additional information than described required actions. We added this additional information to insure the experts would understand what we meant with the capability. However, this had some uncertainties for the evaluation. The complete SPL-capabilities list turned out in a larger than expected document (8 pages only capabilities) which can have demotivating effects on the experts, i.e. large content can demotivate or doing it hastily to finish as quick as possible and not taking their time to do it properly. This in turn can lead to not getting reliable results. We considered narrowing the description of the SPL-capabilities for a more motivating effect. We decided not to do this, since this can make the SPL-Capability less clear and possible the chance to not be comprehended by the expert or misinterpreted. This can also lead to unreliable results.

In structuring the questionnaire we want to present the questions clearly and not give the impression that the questions would be presented randomly. The questions were structured accordingly as was the area where the answer should be inserted (see Questionnaire section). Presenting questions in a not structured manner can comprise the answers, e.g. not presenting the questions in a logical order.

As mentioned before, the evaluation was done abroad. The researcher had an academic contact that was willing to find respondents to perform the evaluation. Since this academic contact would personally contact the respondents we wanted that the questionnaire would be accepted positively on a professional level. This had the advantage of being evaluated by product line experts in a different environment not related to the research or the case company. Hence, we made sure the questionnaire was well-understandable and structured logically.

5.2 The questionnaire

We used a questionnaire for the evaluation for the SPL-capabilities. The questionnaire was intended for practitioners with enough knowledge on SPM, SPL-engineering and SW-development. For this reason the questionnaire stated that it was for '*Practitioners in the role of (or similar) Product (line) manager, Product owner, Product development manager (and/or Project Manager)*'.

Next to evaluating the SPL-capabilities, the added value of the evaluation was to see how existing literature relates to the industry and receive feedback that can be used in addition to the SPL-literature; an view of the industry. In order to prevent misunderstanding, no abbreviations were used in the questionnaire. Furthermore, some questions were further explained or given examples to prevent misinterpretations. The questionnaire consists three sections: General information, Product characteristics and the SPL-capabilities. Below each section is further elaborated.

5.2.1 General information

This section's purpose was to get more information on the expert's experience in the current function and the department. This information will be used for comparison with the case company. The questions are straightforward and are presented below:

What is your function/position within the organization? *(e.g. product manager, product line manager, product owner, project manager, etc.)* How long have you been working in your latest function? *(number in years)* What is the total number of employees working at the department/business unit? *(expressed in FTE, fulltime-equivalent)*

5.2.2 Product characteristics

This section was aimed to get information regarding the product characteristics that the expert's organization is developing. This information is necessary in order to understand in which context the SPL-capabilities are answered and also understand the possible reasoning with respect to the products. The questions used in this section were mostly copied from the SPM Maturity Matrix. As explained in chapter 3, a part of the Maturity Matrix is the Situational Factors list. These are characteristics on aspects such as Business unit, Customers, Market, Product and Stakeholders. As all these characteristics would have been relevant to know and to use for comparison, not all could have been included due to a too large (15+ pages) content to present to the respondents. The Product characteristics were believed to be the most essential ones, since this research is on a product level. In addition, some questions were added, specified on product lines, instead of just products. This was done in order to cover all possible areas that are relevant to the case company, i.e. the (product development) context should be known in order to compare or relate the answers to the case company.

The first two questions are about the production output in terms of product lines and product line members. The following two questions regard the responsibility of the expert towards the products. The more an expert is accountable for, the more overview it has on the management of product line(s). The following four questions are about the age and lifetime of the product line(s) and the members. The last three questions regard the requirements and release frequency of products and the fault tolerance in the products. For instance, we know that the fault tolerance at the case company is very low. This give an impression on how the company deals with product development, i.e. the product has to be completely flawless or not. The questions are presented below.

What is the total number (or estimation) of existing product lines? *(this can thus be one to many)* What is the (average) number of products in a product line, i.e. products constituting one product line? *(this can thus be one to many)* How many product lines are you responsible/accountable for? *(number of product lines)* How many products are you responsible/accountable for? *(number of products)* What is the average age of the product lines?

(Determined by looking at the number of years passed since the first release of the product line until the current point in time. It indicates how long the product line already exists.)

What is the (average) age of products in the product line?
(Determined by looking at the number of years passed since the first release of the product until the current point in time. It indicates how long the product already exists. In this case it can be as old as the product line or younger.)

What is the lifetime of the product lines?
(Determined by the time period the product line will remain in production starting from the current point in time. This indicator thus shows the product line's remaining lifetime, how long the product line already exists must not be included in the calculation.)

What is the lifetime of the products in the product lines?
(Determined by the time period the product will remain in production starting from the current point in time. This indicator thus shows the products remaining lifetime, how long the product already exists must not be included in the calculation.)

What is the release frequency in days?
(Where a release is an update containing functional changes, and not only bugs fixes. For instance, 365 days)

What is the number of new feature request per year from all stakeholders?
(e.g. customers and sales)

What is the tolerance for faults in the products?
(Some products are more sensitive to bugs than others are. If we take for example an application that handles bank transactions then it cannot allow for any defects at all since it could cause grave economical and reputational damage to a business. However, a back office application that is ran only once per week and is non-essential can be non-functioning for a short while without serious consequences.)

5.2.3 SPL-capabilities

This section begins with an introduction text regarding the purpose of the SPL-capabilities. It was made clear that the product management was the main focused instead of the actual development. The structure of the presentation of the capabilities, the business functions, was also made clear. How the capabilities were coded was explained, i.e. a title, a goal and the required action. The last part of the introduction the expert was explained what was expected of him of each capability. Below every capability there is a 'Yes/No' question whether this is a useful capability or not, according to the expert's knowledge and experience. In addition, there was space to give a rationale (fundamental reason) why the capability was useful or not (short or as detailed as the expert chose, however it had to be as comprehendible as possible). Below an example of a SPL-Capability is given as presented in the questionnaire. The SPL-Capabilities were presented as can be seen in Appendix B, in the same order.

A Basic product line scoping

Goal:	Define product line features to support the scope.
Action:	Requirements management defines *Common*, *Variable* and *Product-specific* product features. A Common feature is present in all or most products. A Variable feature is present in some products only. A product-specific feature is present in only one individual product (customer wish). A product feature is a logical unit of behavior that is specified by a set of functional and quality requirements, implying a feature is defined by multiple requirements. By defining the different features, and thus requirements, and in which product they will be present (variable features), the different product line-members can be identified. The aim is to define more common than variable requirements, as variable requirements assure more complexity. However variable requirements are necessary for the overall variability of the SPL.

Is this capability useful?
Yes / No

Rationale:

5.3 Respondents

The researcher performed a research two years ago with a then doctoral student of the Lund University (Sweden) on quality requirements in SPM (Berntsson et al., 2010, 2011). From this collaboration an academic relationship was built and used for this research. This academic contact was willing to find experts with the right prerequisites to fill in the questionnaire. Thus, the respondents whom filled in the questionnaire are experts who work abroad, i.e. Sweden. Our prerequisites were that a respondent should possess enough knowledge on SPM and SPL-engineering by means of their professional experience in the industry. We were interested in the experience experts have in the industry which can be used to evaluate our findings. Next to existing literature it is important to also get feedback from what or how organizations act in the domain of SPM and SPL-Engineering.

Originally, we considered the authors of the SPM Competence Model (Bekkers, W., Weerd, I. van de, Brinkkemper, S.) for filling out the questionnaire. However, we decided (and trusted) to find our respondents abroad and experts who would not have direct relations or even know about the SPM Competence Model. This way, it is not likely their answers will be biased.

Our contact is working at a large mobile organization that has product lines. We requested to find as much as was possible as this would add reliability to the evaluation. However, this was much more difficult than expected. The organization was in the middle of a take-over of another organization and internal re-organizations were taking placing which was keeping all the employees extremely occupied. From all the experts approached, only two at the contact's organization reserved free time to perform the questionnaire. Another respondent at another organization was also willing to complete the questionnaire. In total three experts was found who filled in the questionnaire we sent to Sweden. For confidentiality agreements, we will not elaborate on the details of the experts nor the organizations, since the results can give away sensitive information the organization not wish to be known publically. However, a brief description can be found below.

Organization A
The case company is a large company operating in a market-driven requirements engineering context using a product line approach. The case company has two types of releases, a *major* and a *minor* release. A major release focuses on functionality growth and quality improvements of the product portfolio. Minor releases usually focus on the platform's adaptations to different products. The company has about 5,000 employees and develops embedded systems for a global market. A typical project has around 60- 80 newly added features, from which 700-1000 system requirements are produced. The company has a very large and complex requirements legacy database with requirements at different abstraction levels in orders of 20,000 requirements, which makes it an example of a very large-scale requirements engineering context. A typical project at this company lasts for about 2 years and is implemented by 20-25 teams with about 40-80 developers per team.

Organization B
The company is a global defense company based in Sweden. It has between 5,000 and 15,000 employees and annual sales around 2.7 billion EUR. A typical development project will have several thousand requirements and have a long life span, e.g. 3-5 years. The products are characterized by long time to market due to rigorous requirements on public and operational safety. Development is

usually done in close cooperation with customers, which are in most cases governments or their representatives. The expert was a project manager that mainly works as a developer and a tester. This expert has the most technical experience and knowledge compared to the other two experts given his function. This can also be noticed in his answers, i.e. mostly from a developer perspective. This is what we need for the SPL-capabilities on Architecture.

5.4 The results and modifications

The results of the questionnaire are presented according to the sections made in the questionnaire. We first present the results of the first two sections, i.e. General information and Product Characteristics as these were answered. Afterwards we present the answers on the SPL-capabilities separately. In addition, we also state the needed modification based on the results for each capability. Note that in table 15, Expert A1 and Expert A2 both answered two questions regarding product characteristics with '?' and 'Don't know'. We assume that both answers indicate that they do not have the knowledge to answer the question properly and disregarded their answer.

Table 15: Questionnaire results on the sections of General information and Product characteristics.

Question	Expert A1	Expert A2	Expert B	Case study
General information				
What is your function/position within the organization?	Line manager	Product manager	Project manager (mainly SW developer and tester)	
How long have you been working in your latest function?	2	5	2	
What is the total number of employees working at the department/business unit?	18	60	2500	
Product characteristics				
What is the total number (or estimation) of existing product lines?	1	1	1	
What is the (average) number of products in a product line, i.e. products constituting one product line?	20	15	10	
How many product lines are you responsible/accountable for?	1	1	1	
How many products are you responsible/accountable for?	7	4	2	
What is the (average) age of the product lines?	3	3	10	
What is the (average) age of products in the product line?	3	3	10	
What is the lifetime of the product lines?	?	Don't know	20	
What is the lifetime of the products in the product lines?	12m	12m	20	
What is the release frequency in days?	180	180	190	
What is the number of new feature request per year from all stakeholders?	?	Don't know	less than 20	
What is the tolerance for faults in the products?	Low	Low	Low	

Comparing the expert companies with the case company, we notice that company A also operates in the embedded systems industry. However, at a much higher requirements engineering scale. On the products-level there are similarities; the clearest one being that all companies develop products that have very low fault tolerance. Typically, this implies that these companies must have strong focus on the quality of the end-product. Activities that involve validations are strictly and organized performed in order to assure the successful execution of all sub-processes and artifacts. The case study struggles at this too. The release frequency is a similar aspect. At the moment, the case company has a release frequency of once per year. However, the desire is to have a release frequency of twice per year and last year, two releases were performed for the first time.

The amounts of product constituting a product line also are nearly equal; differing a maximum of 6 or a minimum of 1 product compared to the other companies' business units. However, at the expert companies, multiple practitioners are responsible for the product line with a practitioner being accountable for a product group of 7, 4 or 2 products. Another noticeable distinction is the age of the product lines of the expert companies compared with the case company, making the case company's product line about in the middle. Company A has younger product line whilst Company B has much older one.

Below, we present the evaluation results of two SPL-capabilities as examples. See Appendix C for the complete list of the evaluation of the capabilities. The SPL-Capability in question is represented only by its title due to space saving. The complete capabilities are presented in Appendix B. Furthermore, each capability is presented in the following structure:

- Usefulness: A '3/3' indicates three times a 'YES', whilst a '1/3' indicates one time a 'YES'.
- Rationale summary: This is a summary of what the rationale of the three experts and it supports the Usefulness.
- Modification: Based on the Usefulness and the Rationale summary the needed modification is stated. None means implementation in the SPM Maturity Matrix.

Table 16a: Expert evaluation of a SPL-capability of Requirements Management

SPL-Capability	**Basic product line scoping**
Usefulness	3/3
Rationales summary	It is important to know if the feature/functionality is variable before development start. Literature suggests that in order to build in the variability properly, it has to be known in an early phase.
Modification	None

Table 176b: Expert evaluation of a SPL-capability of Requirements Management

SPL-Capability	**Advanced product line scoping**
Usefulness	1/3
Rationales summary	It is not useful when fewer and larger variabilities are at stake. This means that it is useful for many and smaller variabilities. However, it is useful to detect early hazardous behavior of the system (similar of the focus area Release build validation).
Modification	This capability needs a condition: useful IF variability in the product line is large in number and it is relatively small variation.

5.5 Summary

The evaluation proved to be helpful, despite the resources limitations. The SPL-capabilities were evaluated by expert with knowledge and experience regarding SPM and SPL-Engineering. Regarding the questionnaire, we did not receive critique or negative response. This might have been different if more than three experts would have completed it. Furthermore, we did not receive signs of misinterpretation regarding the questions. On two questions we did notice that two expert did not know the answers to, answering with '?' and 'Don't know'. Regarding the capabilities, one expert clearly specified that he had no experience with product line architectures. Here the answer of the expert with most technical experience weighted more. In total, out of the 21 SPL-capabilities that were evaluated, nine (9) did not need modifications, four (4) will be left out and eight (8) needs modifications.

6 Software Product Line Management Maturity Matrix

In this chapter we will implement the SPL-capabilities presented, discussed and evaluated in the previous chapters, in the SPM Maturity Matrix. The aim of this chapter is to get the Maturity Matrix 'SPL-ready' in order to be applied at the case company. Two major steps will ensure this: first, the SPL-capabilities will be rewritten properly, and second, the SPL-capabilities will be given a maturity, i.e. a placement in the matrix. After these steps the first version of the Software Product Line Management Maturity Matrix (SPLM Maturity Matrix) is born.

6.1 Capabilities revision

So far, the SPL-capabilities have been identified from the SPL-practices, mapped to the SPM Maturity Matrix and evaluated by experts abroad. However, we still do not have a Maturity Matrix with SPL-capabilities. In this section we will firstly apply the needed modification based on the evaluation results (section 5.4). We reflect on the attributes Title, Goal and Action and revise where needed.
In addition, we will rewrite each SPL-Capability based on its essence; this implies that the additional information is separated from the Action-attribute of the capability to the Rationale-attribute (see section 5.1). Finally, the corresponding references are added. Below we present the capabilities categorized per business function and focus area.

Requirements management

Requirements gathering

Title	Basic product line scoping
Goal	Define product line features to support the scope for product line development.
Action	Requirements management defines Common, Variable and Product-specific product features. A Common feature is present in all or most products. A Variable feature is present in some products only. A product-specific feature is present in only one individual product (customer wish).
Reference(s)	Pohl et al. (2005), Bosch(2002), Taborda (2004), Schmid et al. (2007)
Rationale	A product feature is a logical unit of behavior that is specified by a set of functional and quality requirements, implying a feature is defined by multiple requirements. The aim is to define more common than variable requirements, as variable requirements assure more complexity. It is important to know if a feature is variable before development starts (early phase) to ensure the variability is built in properly.

Requirements identification

Title	Advanced product line scoping
Goal	In-depth analysis of commonalities and variability
Action	Example/real-life scenarios are deployed for further analysis and evaluation of the commonalities and variabilities. Usability scenarios describe actions required to perform common user operations. Variability scenarios emphasize the differences between individual products. In cases where differences are difficult to identify, prototyping makes it possible to explore profound differences between the members. CONDITION: *Implement IF variability in the product line is large in number and it is relatively small variation*
Reference(s)	Ardis et al. (2000)
Rationale	Prototyping is vital in cases where a product line is developed, specifically for a niche market or customer.

Requirements organizing

Title	Components dependency registration
Goal	Record the dependency of the components
Action	Determine and register components' dependencies. A dependency exists when a component

	requires that another components be implemented too (or specific actions) for it to function properly or in cases of conflicts, for it not to be implemented. Components' dependencies are described textually and/or modeled. Modeled description offers better communication. CONDITION: *the more products this capability is applied to the greater the benefit will be*
Reference(s)	Jaring et al. (2002)
Rationale	These dependencies are direct input for the architecture. Components' dependencies should be described when the corresponding features are organized accordingly. This gives great benefits when the impact of changes on components is being analyzed. However, according to practice very hard to maintain.

Title	Product line requirement life cycle management
Goal	Make requirements traceable through detailed information.
Action	Register in which member the components will be implemented in. This start with requirements constituting features, features being bundled into components and components being developed and (re)used to build the end-product. The registration of the requirements during the whole trajectory is important. Detailed information such as, requirement submitter, date, status (new, verified, planned for release x.y, in-progress, tested, completed, etc.), description, etc. which was not known before, should be added to the requirement, feature or component.
Reference(s)	Linden et al. (2002), von der Maβen et al. (2004), Rombach (2005)
Rationale	This is necessary in order to be able to trace requirements down (history of requirements implementation) and have the knowledge of how an issue has been solved before. It should be clear which SPL-members should get updated versions of core components when these are available. The value of this capability is seen when it is time to test/validate the product. This way the requirements performance is clear.

Release planning

Requirements prioritization

Title	Components consideration
Goal	Considering components when prioritizing product line features
Action	Prioritize the features that will be implemented in end-product from the next release on, with the intended components in mind. The prioritization is performed with the components in focus and which features are required. This implies that features would be implemented in the end-product, through components.
Reference(s)	Taborda (2004)
Rationale	Prioritization is necessary, since not all features can be implemented, due to resources and delivery deadlines. Each requirements/feature should have a business value, implementation costs and architectural implication. The more a feature or component is shared in more products the higher market value it should get.

Release definition

Title	Product line features selection
Goal	Selection of features for implementation for the next release.
Action	A practical selection of the features is made given the limitations on engineering resources, based on the priority assigned. The selection is defined textually. The function and essence of the components is also considered when making the selection. Business priority should not always be dominant in the selection. Technical benefits and limitations should also be considered.
Reference(s)	Taborda (2004), Pohl et al.(2005)
Rationale	The function and essence of the components is also considered when making the selection, i.e. which features are of importance for the sake of the component. Business priority should not always be dominant in the selection.

Title	Product line release planning

Goal	Release plan for product line components
Action	A release plan is created for the components. Release dates are determined for the components, supported by detailed information on the features. Usually these are core components which will be reused through the platform. Do not plan too much ahead, since this lets the resistance of having variability in the products gets too high.
Reference(s)	Taborda (2004), Pohl et al.(2005)
Rationale	Prioritized features are organized into components (e.g. sharing same functionality), the release of the components can be planned. This can be for market introduction and/or for specific customers. The reason behind this is that, in product lines components are reused for future product releases (part of maintaining the platform). However, reuse has to have added value in order to take place. Do not plan limitless reuse.

Release build validation

Title	Architectural release validation
Goal	Release validation by architecture
Action	The design and coding of the SPL-members (the build) is validated through the product line architecture before the actual release is launched. The design and software code have to adhere to the limitations and structure of the product line architecture. This validation is performed by the department(s) who is (are) responsible for developing and maintaining the product line's architecture(s).
Reference(s)	Taborda (2004), Pohl et al.(2005)
Rationale	The product line's architecture is vital for achieving the business goals set up front, when management decides to engage in SPL-Engineering. This should be performed during development, since if a mistake is detected then it rarely happens that a market release will be delayed because the code is not following internal standards. If the architecture is not followed, other SW development activities might be affected, e.g. tests modules might not function properly or certification of the source code will be challenging, because constraints in the architecture might be invalidated.

Product planning

Product roadmapping

Title	Product line roadmap
Goal	Roadmap creation to define the product line scope
Action	A roadmap is created detailing the anticipated products of the product line, as far as foreseeable. The components that constitute the members are also presented (if possible), i.e. multiple features form a component. Product line features should be predicted for approximately 2 years in the future, including product or component releases.
Reference(s)	Svahnberg et al. (2000), Jaring et al. (2002), Linden (2002), Pohl et al. (2005)
Rationale	According to some literature, the product line features, common and variable, should be predicted for a time span of 5 years. However, other authors (e.g. Svanhberg et al.) believe this is not practical, since a great amount of the future requirements (aside from technology shifts and/or other development changes) of the product line can hardly be predicted for longer than 1 year into the future (implying that features defined today will not feed the markets or customers' needs over a 5 year span).

Core asset roadmapping

Title	Infrastructure standardization
Goal	Standardize the infrastructure for developing core assets
Action	Standardize the development infrastructure where the product line members are created. The infrastructure ensures certain degree of product quality and usually exists of an operating system combined with commercial components on top such as database managements systems, integrated development environment, graphical user interface, etc., all which are needed to develop and maintain core assets in a SPL environment.
Reference(s)	Linden (2002), Böckle et al. (2005)

Rationale	Increasing commonalities will need to be managed in order to be exploited properly and allow flexibility. The managed commonalities are developed into fundamental components (core assets). These core assets mostly contain domain (generic) functionality. This capability is one of the first steps organizations should take when exploiting commonalities in its products

Title	Platform introduction
Goal	Intra-organizational reuse of core assets – exploiting core assets.
Action	Develop, maintain and evolve a development platform which allows intra-organizational reuse of core assets. This implies SW-code such as drivers or interfaces, but also documentation or release notes. Features that are shared by sufficient members are included in the core assets through the platform, whereas features shared by only few members are developed as part of product derivation.
Reference(s)	Linden (2002) , Böckle et al. (2005)
Rationale	In addition to the previous capability, this one ensures knowledge sharing and more efficient product development as the organization is more involved in development and uses the platform on an organizational level. The drive management has behind such a platform is the fact that profit levels can increase if performed correctly, i.e. development costs decrease as core software is developed only once and reused multiple times. Important to protect and clearly divide development responsibility here. Make sure that product development (AE) does not burden core assets development (DE) too much. This can lead to a negative effect on the core assets development team by constantly switching of context which will lead to a decrease of productivity.

Roadmap intelligence

Title	Visualization of the product line scope
Goal	Visualize product line scope for communication with stakeholders
Action	The members and their features are represented in a product map, i.e. a matrix with the members in the columns and the features in the rows as well as information on the market, costs, competitors and benefits (not date nor schedule is included, not to be mistaken it with the product roadmap). This product map is communicated with e.g. management, sales, services, customers, suppliers.
Reference(s)	Bayer, et al. (1999)
Rationale	The product line scope is presented with all valuable information in one overview. It is possible that organizations use other ways to visualize the same type of information. This is also a way to represent traceability; to allocate features to members together with other valuable information for the product line. Developers are interested in the details such as features and requirements, whilst product planners are also interested in market information and competitors.

Portfolio management

Product lifecycle management

Title	Financial Product line scoping
Goal	Product line scoping with financial information
Action	The decision whether a product will or will not be part the product line scope is based on the expectations of the ROI and other possible financial calculations. Financial information that will add value to the decision-making is included to clarify the financial impact the products may have. If these are beneficial for the organization, the product will be part of the line, otherwise it will be declined. Same goes for features, i.e. dropped according to their expected added value or revenue.
Reference(s)	Linden (2002), Schmid et al. (2007)
Rationale	This is useful for features. This also assures a sort of cleaning up in the requirements collection and code, since requirements are left out or removed if these do not bring financial added value.

Not coupled to any business functions

Product line architecture

Title	Basic reference architecture construction
Goal	Create product line reference architecture with requirements input
Action	This architecture creation is driven by the functional requirements (generic scenarios; which are similar to commonalities) and/or domain-independent quality aspects (property-related scenarios; which are similar to variable features with a quality focus). The generic scenarios are combined with property-related scenarios and rated according to architectural importance. Iteratively, functional requirements from the generic scenarios are chosen to create the initial architecture. In each iteration, other property-related scenarios are added to the candidate architecture to complete it and refine it. These 'scenarios' can be seen as guidelines as other techniques also possible. Mostly, requirements (common, variable and product-specific) are used to initiate the architectural design of the software product line,
Reference(s)	Bayer et al. (1999), Schmid et al. (2007)
Rationale	The aim is to define a domain-specific architecture that covers the existing products and future SPL-members. Concepts are determined and modeled according to their relation and product line scope for the architecture creation initiation. The property-related scenarios are used to validate and refine the candidate architecture

Title	Reference architecture construction
Goal	Create product line reference architecture based on components
Action	Create the product line reference architecture. Component frameworks are used to support the various types of quality requirements, i.e. frameworks are used to model features in a structured manner into components with their relationships. The reference architecture has to solve issues of variability and reusability. It also properly incorporates quality requirements such as flexibility, maintainability and evolvability at an early phase.
Reference(s)	Svahnberg et al. (2000), Bosch (2000), Pohl et al. (2005)
Rationale	Role of the architecture is to describe the features of the products in the product line and to provide the overall structure. Common and variable features, represented by the components, are the main drivers for architectural design together with quality requirements such as performance, security, usability, etc. The earlier common and variable features are considered, the more flexibility is assured for the product line. As features are represented in components, the components in the architecture implement a particular/coherent domain of functionality, e.g. the network communication domain.

Title	Advanced reference architecture construction based
Goal	Product line architecture development driven by requirements management
Action	Commonalities and variability are modeled in the architecture. Variability in the requirements is modeled through variation points and variants. The architecture defines the components, mandatory, optional, and alternative, component interrelationships, constraints, and guidelines for use and evolution in building the product line. Quality predictions need to be considered early on. The most important architectures issues that have to be addressed properly are: Significant architecture requirements (functional requirements & quality requirements), Concepts (the architecture concepts clarifies the architecture organization), Texture (standard solutions for implementation problems) and Structure (internal organization of the products).
Reference(s)	Linden (2002), Schmid et al. (2007)
Rationale	The design for the reference architecture for the product line receives input from Requirements Management. The architecture should relate all design decisions made, to the requirements. Variability modeling supports quality predictions. The risk if variability is not considered is that the usefulness of the architecture might be significantly lower than desired. It will limit the number of possible variants and products and increase modification costs.

Organizational

Title	Requirements engineering planning
Goal	Plan requirements engineering for product line developments
Action	Product management controls Requirements engineering directly, first on domain level afterwards on product (or application) level. Common and variable features of the product line are defined in the roadmap, which serves as the scope for further requirements engineering tasks such as detail specification and modeling. Afterwards, Product Management defines which products or applications, implying which requirements will be implemented for which product, individually (variable features)
Reference(s)	Ardis et al. (2000), Deelstra et al. (2003, 2004), Pohl et al. (2005), Schmid et al. (2007)
Rationale	It is important in order to know what to implement in which product and you need to plan this instead of letting everyone implement what he thinks is important.

Title	(Upper) Management Involvement
Goal	Proper execution of product line processes
Action	(Upper) Management is actively involved and monitors the processes of SPL-development that concerns delivering reusable core assets and the quality thereof. Furthermore, management is also responsible for focusing on the best fit organizational structure for the organization.
Reference(s)	Northrop et al. (2000)
Rationale	This is for guidance on keeping the processes on track and not to drift away from the business goals. (Upper) Management is more involved in the above mentioned processes to keep, e.g. by monthly reports or management involved, as a stakeholder, at the end of each Sprint when following agile Scrum. However, this capability is relevant for the overall success of SPM processes.

6.2 Capabilities positioning

In this section we will implement the SPL-capabilities in the SPM Maturity Matrix. This implies that the capabilities presented in the previous section will be positioned appropriately. When performing this positioning, each SPM-Capability is analyzed and compared to the SPL-Capability to assess whether the SPL-Capability is positioned best given the business function and focus area determined in previous sections. The reasoning supporting the SPL-Capability position (the given maturity) is given in the field 'Rationale'. SPM-capabilities that are not mentioned are automatically applicable for product lines. Unfortunately, due to time limitations, we did not get to evaluate the SPL-capabilities positioning. This would have been of added value, since now the positioning is done based on the researcher's knowledge and experience. The expert evaluation gives a helping hand in this. The result of the positioning will be presented in an overview of the complete SPLM Maturity Matrix.

Given the definition of the Product planning business function by Bekkers et al. (2010), we decided to add a focus area to this business function, i.e. the *Product line architecture* focus area since it will be a group of coherent architecture capabilities. '*Product planning is concentrated around the gathering of information for, and creation of a roadmap for a product or product line, and its core assets*', implies that information for and the creation of a roadmap is central. However, this is not completely the case for Product line architecture. For Product line architecture the roadmap is essential for the reason that in a SPL-environment the roadmap would include specifics on common and variable features of the product line. These specifics features are direct input needed for the creation of the architecture. In addition, the other three business functions do not give a better placement for Product line architecture focus area. Requirements management is concentrated on the requirements themselves not regarding a release or more. Release planning is focused on creating and launching a release successfully, which is not on a product line level. Portfolio management is concentrated on the strategic information gathering for decision making across the entire product portfolio. The focus area Product lifecycle management mentions product lines, however this is a

capability on a portfolio level which is 'higher' than product line architecture. If performed properly, according to literature, a link can be made; when the SPM product line capability (PLM:E) is implemented, at least one Product line architecture capability should be implemented too.

Most SPL-capabilities are an extension of a SPM-Capability, i.e. the SPL-Capability requires somewhat extra actions or aspects to be considered. Below, we present the SPLM Maturity Matrix and after that the positioning elaboration for each SPL-Capability. Note, a SPL-Capability is presented in **blue**. A '*' behind a letter indicates a SPM-Capability with an increase in letter (however with its original maturity). This is to prevent capabilities to have same letters for one focus area. A SPM-Capability that is moved (increased) to another maturity is presented in **green**. In this case, a SPL-Capability is given slight priority over the SPM-Capability. A green **'*'** indicates a SPM-Capability that has both an increase in letter (e.g. from 'D' to 'E') and an increase in maturity (e.g. from '5' to '6'). Usually, these come after a SPL-Capability.
For instance, capability **B** of Requirements gathering is a SPL-capability. Since this capability is positioned in front of the other SPM-capabilities, the other SPM-capabilities are 'pushed' one letter further to prevent having capabilities with the equal letters.
Capability **C*** of Release definition had to be re-positioned to a greater maturity in order to make place for the SPL-capability of **B** (in front of Release definition:C). Hence, the capability is green. The same capability also has a '*', for it had to be 'pushed' a letter further also to prevent capabilities with equal letters.

Process \ Maturity	0	1	2	3	4	5	6	7	8	9	10
Requirements management											
Requirements gathering		A	B	C*	D*		E*	F*	G*		
Requirements identification			A	B		C*		D*			E*
Requirements organizing				A		B	C	D*	E		
Release planning											
Requirements prioritization			A	B	C*	D*	E*			F*	
Release definition			A	B	C*	D	E*		F*		G*
Release definition validation					A			B		C	
Scope change management				A		B		C		D	
Build validation					A			B	C	D*	
Launch preparation		A		B		C	D		E		F
Product planning											
Roadmap intelligence				A		B	C		D	E	F*
Core asset roadmapping					A	B	C*		D*	E	F*
Product roadmapping			A	B	C		D*	E*		F*	
Product line architecture				A				B		C	
Portfolio management											
Market analysis					A		B	C	D		E
Partnering & contracting						A	B		C	D	E
Product lifecycle management					A	B		C	D*	E*	F*

Figure 2: The Software Product Line Management (SPLM) Maturity Matrix.

Requirements management

Requirements gathering B

Title	Basic product line scoping
Maturity	B2
Prerequisite(s)	Requirements gathering A
Position Rationale	No prerequisites. This is the most basic and the first capability an organization should implement in order to handle requirements/features properly for product line development at a very early phase. As soon as requirements are gathered this capability can take place.

Requirements identification B

Title	Advanced product line scoping
Maturity	B3
Prerequisite(s)	Requirement gathering B, Requirements identification A
Position Rationale	Due to prerequisites it can't be 1 nor 2. The further identification of commonalities and differences is needed if uniformity of requirements (RI:A) was not successful in achieving this. Commonalities and differences are a vital focus point for SPL.

Requirements organizing C

Title	Product line requirement life cycle management
Maturity	C6
Prerequisite(s)	Requirements gathering B
Position Rationale	This capability is similar to that of SPMCM RO:B5. However, for product lines the focus lies more on the trajectory of requirements forming features, and features forming (core) components, and (core) components being built in order to be used during product development. Within this trajectory, traceability is of great importance. It is directly placed after RO:B, since it is an extension of the actions performed in RO:B.

Requirements organizing E

Title	Components dependency registration
Maturity	E8
Prerequisite(s)	Requirements organizing C
Position Rationale	This capability is an extension of the SPMCM RO:C7(RO:D7). The focus for product lines lies not on the requirements level. It lies on the components level. Hence the prerequisite. Component dependency has also been pointed out to be important for product lines in literature, e.g. for product line architecture

Release planning

Requirements prioritization B

Title	Components consideration
Maturity	B3
Prerequisite(s)	Requirements gathering A
Position Rationale	Some consideration of product line components have to take place before this capability can be implemented. Hence the prerequisite. Right after prioritization took place with internal stakeholder, product line components should be considered.

Release definition B

Title	Product line features selection
Maturity	B3
Prerequisite(s)	-
Position Rationale	This capability is similar to the SPMCM RD:A. However, it is more advance since components need to be considered (or are mandatory) when making the selection of the features for product lines. As a consequence of the capability positioning, the SPM-capabilities of RD:Standardization and RD:Internal communication had to be moved one maturity greater respectively.

Release definition D

Title	Product line release planning
Maturity	D5
Prerequisite(s)	Requirements prioritization B
Position Rationale	This capability fits when a release definition has been made and needs further specification on components. Release dates can be easier determined when features selection is known. This release plan has to be created before being internally communicated.

Release build validation C

Title	Architectural release validation
Maturity	C8
Prerequisite(s)	Product line architecture A
Position Rationale	The prerequisite indicates that a SPL-architecture needs to be in place. Before the Build goes through external validation (RBV:Certification), it has to be validated internally by the architecture. SPL-members have to adhere fully to the architecture in order to be successful. Validation through the architecture ensures that the members are built the way they were intended to with regards to components dependency, constraints, interfaces, variabilities, etc.

Product planning

Product roadmapping C

Title	Product line roadmap
Maturity	C4
Prerequisite(s)	Release definition B
Position Rationale	With the selection of features and components known, the roadmap can be created (prerequisite). For product lines commonalities, variabilities and components are included in the roadmap. Due to the rapid changes in technology nowadays and the knowledge of the SPM-capabilities we decided to put this to 2 years.

Core asset roadmapping B

Title	Infrastructure standardization
Maturity	B5
Prerequisite(s)	-
Position Rationale	After the SPM-Capability of registering and storing all core assets, this capability provides the ideal development and management infrastructure for core assets. This allows a more efficient way of developing and provides the possibility for the organization maintain quality of the assets. In addition, this capability also support the other SPM-capabilities such as 'Core asset identification' and 'Make or buy decision', since through the infrastructure core asset information is easily obtained and managed.

Core asset roadmapping E

Title	Platform introduction
Maturity	E9
Prerequisite(s)	Core asset roadmapping B
Position Rationale	This capability further extends Core asset roadmapping B onto a more organizational level. Before being able to construct roadmap for core assets, this capability gives more certainty (input) on the development (evolution) of core assets and their future use. Once this certainty is accepted, roadmaps of core assets are more reliable.

Roadmap intelligence E

Title	Visualization of the product line scope
Maturity	E9
Prerequisite(s)	Roadmap intelligence A, (B & C) Roadmap intelligence D
Position Rationale	This capability is placed right after its prerequisites and right before the most mature capability of this focus area 'Partner roadmap'. It is mainly a combination of the capabilities in the prerequisites, in one overview. The positioning has the consequence that the capability 'Partner roadmap' moves to F10.

Product line architecture A

Title	Basic reference architecture construction
Maturity	A3
Prerequisite(s)	Requirements gathering B

Position Rationale	As soon as common, variable and product-specific features are known, a start of the product line architecture can start. These and quality requirements are the main input necessary for the construction of the architecture.

Product line architecture B

Title	Reference architecture construction
Maturity	B7
Prerequisite(s)	Product line architecture A, Requirement organizing C
Position Rationale	This capability requires a more understanding and structure in requirements in order to take place. Once the requirements have been organized, (future) components are known this capability can be implemented

Product line architecture C

Title	Advanced reference architecture construction based
Maturity	C9
Prerequisite(s)	Product line architecture A, Requirements organizing E
Position Rationale	At a more advanced stage in architecture construction, all possible information on the functionalities, quality requirements, commonalities, variabilities, components, dependencies is required to construct the reference architecture properly. Due to the prerequisite this capability has the maturity of C9.

Portfolio management

Product lifecycle management C

Title	Financial Product line scoping
Maturity	C7
Prerequisite(s)	-
Position Rationale	Before the capability of 'Portfolio scoping analysis' this capability provides a financial scoping for product line members and/or features. The results of the scoping provide more information and better decision-making on the capabilities that follows, i.e. both on product level as on feature level.

Organizational supportive input

The organizational capabilities presented in the previous section, i.e. *Requirements engineering planning* and *(Upper) Management involvement* are not included in the maturity matrix. We decided not to position these with the capabilities due to that they are less detailed capabilities for product lines with respect to SPM-practices compared to the other capabilities. In addition, these two capabilities are more applicable on an organizational level, i.e. their actions are performed in order to organize and create structure for the product management practices. However, their importance shall not go unnoticed. Therefore, we decided to name these as Organizational supportive input that are *not* included in the SPML Maturity matrix, however they have to be acknowledged.

Requirements engineering planning mainly points out that one department should be responsible for the coordination of product management. This is important for making key-decisions, a controllable process and an environment where practitioners can function effectively and efficiently. If this is not clear within an organization, product management cannot be exploited since there is less structure and support for practitioners and decisions will have to be made ad hoc without proper preparation or proper information.

(Upper) Management involvement has no direct implication on product management practices. Usually, it is the decision of upper management to engage in product line development. The same management is expected to be involved with product management in order to monitor the greater milestones and deliverables with the purpose not to deviate from the business goals.

Title	Requirements engineering planning
Goal	Plan requirements engineering for product line developments
Action	Product management controls Requirements engineering directly, first on domain level afterwards on product (or application) level. Common and variable features of the product line are defined in the roadmap, which serves as the scope for further requirements engineering tasks such as detail specification and modeling. Afterwards, Product Management defines which products or applications, implying which requirements will be implemented for which product, individually (variable features).
Reference(s)	Ardis et al. (2000), Deelstra et al. (2003, 2004), Pohl et al. (2005), Schmid et al. (2007)
Rationale	It is important in order to know what to implement in which product and you need to plan this instead of letting everyone implement what he thinks is important.

Title	(Upper) Management Involvement
Goal	Proper execution of product line processes
Action	(Upper) Management is actively involved and monitors the processes of SPL-development that concerns delivering reusable core assets and the quality thereof. Furthermore, management is also responsible for focusing on the best fit organizational structure for the organization.
Reference(s)	Northrop et al. (2000)
Rationale	This is for guidance on keeping the processes on track and not to drift away from the business goals. (Upper) Management is more involved in the above mentioned processes to keep, e.g. by monthly reports or management involved, as a stakeholder, at the end of each Sprint when following agile Scrum. However, this capability is relevant for the overall success of SPM processes.

7 Case study

CASE STUDY REMOVED FOR CONFIDENTIALITY REASONS

8 Conclusion, Discussion and Future research

In this chapter we will present the conclusion of this research, the limitations and possible future research directions. For the conclusion we will use the research findings and results to answer the research question presented in chapter 2. Next, we will discuss the limitations of this research through the applied methodology and results. Finally, we will also present future research directions leading from this research.

8.1 Conclusion

The conclusion of this research is drawn by answering the research questions presented in chapter 2. Below, we restate the main research question:

"How can product software companies structure their software product management processes according to their product line development approach?"

For answering this question, first the topic of Software Product Lines had to be studied in order to know the characteristics of this field of practice. This study identified several SPL-practices that are essential and have influence on SPM processes. For SPM we used a known, validated and already applied in several product software companies competence model for the fundamental knowledge of SPM; the Software Product Management Competence Model and corresponding Maturity Matrix (Bekker et al., 2010). The SPL-practices were processed to an equal level of the maturity matrix and through validation the essences of the SPL-practices were implemented in the maturity matrix, making it into the Software Product Line Management Maturity Matrix (SPLM Maturity Matrix). The SPLM Matrix is a useful assessment instrument for product software companies that develop according to a product line approach. It presents the most important SPM-processes in a structured and hierarchical manner and, in addition it also presents the specifics that are essential for software product lines. The answer to the main research question is possible through answering the sub-research questions which will further explain, and are presented below.

1. *Which key practices can be identified in software product line developments, from a Software Product Management perspective?*

   The answer to this question is presented in section 4.1. In particular, table 11 gives an overview of the 12 identified SPL-practices, below we restate the practices:
   - Domain & Application Engineering
   - Core asset, Product development & Management
   - Variability management
   - RequiLine
   - Product line Release planning
   - Product Portfolio Planning – QFD
   - PuLSE
   - Product Derivation Framework
   - Integrated SPPL
   - SPL FAST Process
   - BAPO
   - BAPO Evaluation Model

   These identified SPL-practices are the result of an extensive literature study (see chapter 3). In performing this literature study, we came across various SPL related literature from which the

SPL-practices were identified, i.e. some SPL-practices were identified in multiple literature and vice versa. We used the SPM Competence model (Bekkers et al., 2010) as our SPM foundation in order to recognize when a SPL-practice is related to product management. Some SPL-practices were clearly related while others were at first not (enough) related to SPM. Hence, the distinction was made between SPL-practices of 'Product Management' and 'Development'. In addition, we further analyzed these product line key practices to identify the activities that combined form each practice; these are SPL-activities. From the 12 SPL-practices, 84 SPL-activities were identified. The SPM Competence model was also used to distinguish which SPL-activity was SPM related and which not. In table 11 we present a shorten version of the SPL-activities. See Appendix A for the complete overview of the identified SPL-activities with the corresponding SPL-practice.

2. *What different Software Product Line development situations can be distinguished that influence product management processes?*

   From our literature study, we struggled to find relevant researches regarding the influence of different product line development situations on SPM processes; most researches were directly too project management related or too business management related. One research covered development projects on a product line level (platform project) and on a product variance level (derivative project) what is similar with CCV. Unfortunately, this research further focused on project management; project task characteristics, project planning, project execution and success.

   The intention of this sub research question was to identify probable situations *when* development will be initiated and how these situations are best organized from a SPM perspective. No limitations were made on these development situations (e.g. only 'projects'), i.e. it was not obliged to organize the SPM processes for a development situation, context or cycle according to and naming it a 'project'.

3. *How can the obtained knowledge be used to structure Software Product Management for product line developments?*

   To answer this research question we used the results of the first sub research question and the SPM Competence model. The SPM Competence model and maturity matrix is primarily used to determine the maturity of a PSC by assessing the essential business functions of the SPM practice through in-groups-defined capabilities (focus area's). Secondarily, the maturity matrix is used to give improvement recommendations through incremental steps. By adapting the SPM Competence model or following its recommendations, a PSC organizes its processes according to the structure maintained by the competence model (see section 3.1), for instance, following the four main processes Requirements management, Release planning, Product planning and Portfolio management.

   Sub research question 1 resulted in identified key practices of SPL developments with respect to product management. The results were further analyzed to identify the activities performed that characterize each practice. By joining this knowledge accordingly into SPM maturity matrix, the overall SPM knowledge is enhanced with product lines specifics and a validated structure is maintained. In other words, the gain SPL knowledge is presented in a structured manner (that of the SPM Maturity matrix).

   Joining the SPL knowledge in the SPM maturity matrix was performed in the mapping process (chapter 4). The identified SPL-activities were analyzed and compared with the SPM-capabilities of

the SPM maturity matrix. A mapping implies a defined status for the comparison with the SPM maturity matrix; a Similarity, Neutral or a Candidate improvement. In addition, a mapping registers to which part of the matrix the SPL-activity is most related to, i.e. business function, focus area, and/or SPM-capability. Candidate Improvement means that the SPL-activity adds SPL specific information to the matrix at that particular area (i.e. following the hierarchical structure, particular business function, focus area and/or SPM-capability).

Next, the Candidate Improvements (SPL-activities) are re-defined according to the same structure as of the SPM-capabilities. This structure implies that, inter alia, a title, a goal, an action and references must be defined for the SPL-activity. Once a SPL-Activity is re-defined in this structure, it becomes a SPL-Capability (see section 4.3).

4. *How applicable is the Software Product Management Competence Model for software product line development?*

The answer to this question lies, partially, in the results of the mapping process. From the 84 identified SPL-activities, five were similarities, 11 were neutral-organizational, 12 were neutral-architecture, 18 were neutral and 38 were candidate improvements. As can be seen, only five SPL-activities could be declared similar (practically equal) to the SPM maturity matrix. The largest majority of SPL-activities, 41, are neutral. These neutral SPL-activities do not often describe information that can directly be related to the maturity matrix, e.g. most neutrals describe SPL-activities on an architectural and/or organizational level, which are essential for SPL development. However, these SPL-activities (neutral) do not indicate a positive applicability towards the SPM maturity matrix since both architectural and organizational activities are not included in the matrix. In fact, it is these neutral SPL-activities that did not resulted in any business function of the SPM maturity matrix after they have been defined as SPL-capabilities. Eventually, three SPL-capabilities (architecture) were formed and successfully implemented into the maturity matrix from 6 of the 41 neutral SPL-activities. The organizational SPL-activities did not result in any SPL-capability, however we defined two of the most important activities as 'organizational supportive input' from a SPL development perspective (see section 6.2). For the SPL-activities regarding product line architecture and organization, the SPM maturity matrix resulted to be less applicable as for SPL-activities that were similarities and/or candidate improvements.

In the mapping process, the positioning of the SPL-capabilities in the SPL maturity matrix is determined. This positioning implies giving the SPL-capabilities a maturity within the matrix. Most of the SPL-capabilities did not replace the maturity of a SPM-capability. Only the focus area's 'Release definition' and 'Roadmap intelligence' required SPM-capabilities (3 from 68) to be replaced, i.e. increase of '1' in maturity. This has no further impact. The other SPL-capabilities were successfully implemented in 'empty' cells of a maturity in a focus area.

From a SPL-capability essential perspective, the expert evaluation resulted in 4 SPL-capabilities being left out and 8 needing modifications. This indicates that 16 of 21 SPL-capabilities have enough essence (valuable knowledge) according to the experts from the industry to be applicable in real life. Furthermore, no significant difference was identified or mentioned during the second interview sessions where the SPL-capabilities were discussed other than the subjects of the discussions. This is an indication that the implemented SPL-capabilities in the maturity matrix were experienced in similar as the SPM-capabilities.

We conclude that the SPM maturity matrix (and competence model) is applicable for SPL development; on the same granular level we can state that it is 68 (total SPM-capabilities) / (68 (SPM)+16 (SPL)) * 100% = 81% applicable for SPL development, i.e. the SPM-capabilities can be applied to SPL development without it being completely irrelevant and thus waste time. The other

19% is pure SPL specific information that has been implemented in the matrix in order to apply better to SPL development.

5. *How can Product Software Companies use the gained knowledge to improve their Software Product Line Development approach?*

   The gained knowledge after this research is mostly presented in the SPLM maturity matrix, through the SPL-capabilities. Not all gained knowledge of this research is implemented in the matrix; however, some of which still is useful for SPL development. Below, we state the gained knowledge and elaborate how it can be used for improvements:

   SPLM Maturity Matrix
   The essence of the SPLM maturity matrix still remains as an assessment method of SPM processes. It has been specified for PSC's that have a SPL development approach. PSC's can use the SPLM maturity matrix to assess and know the maturity of their SPM processes in their SPL environment. Furthermore, the SPLM maturity matrix provides incremental improvements, depending on the business strategy of the PSC.

   Organizational Supportive Input
   Some of the SPL-capabilities that were not implemented into the matrix, are capabilities on a more organizational level of product management. The actions of these capabilities have the purpose to benefit the organization of the SPM processes in particular.
   Firstly, *Requirements engineering planning* should be primarily and mainly controlled by Product Management (see section 6.2). This is important for an overall controllable process and an environment where requirements are dealt with accordingly to SPL development; Product management is responsible for this. Secondly, *(Upper) Management* is expected to be involved in the SPL development through the monitoring of milestones, main deliverables and keep development towards the business goals. We have noticed that where management involvement is low, SPL success is less probable.

   Communication
   From the performed case study, we have learnt that communication of the right information is of great importance in SPM and product line development (see section 7.5). Stakeholders of the product management processes should share information gained regarding their given responsibility. Particular information can be notes, official documents, agreements, presentations, roadmaps, planning, budget statement, test results, release definitions, requirements selection, product feature groups, etc. This should work from practitioners responsible for the product portfolio to the practitioners responsible for coding the software.

   Product Line Architecture
   Another great aspect of importance we learnt from the case study is product line architecture. When performed properly, SPL development adheres to the reference architecture. This reference architecture should be sustainable for the expected life time of the product line and expandable to certain degree. In developing this architecture, PSC's should pay extra dedication on implementation of common and variable features, quality requirements and development platform support (see section 7.5) in the reference architecture.

8.2 Discussion

The main artifact that his research has delivered is the SPLM maturity matrix. Limitations exist in the implementation of SPL knowledge into the SPM maturity matrix. Firstly, the SPL-capabilities need validation from the SPM Competence experts. The SPL-activities have been defined as SPL-capabilities by the researcher, exactly according to the standards of SPM-capabilities. However, since the researcher is not an original author of the SPM Competence model, it adds value to the defined SPL-capabilities if experts of the SPM Competence model could validate them. For instance, some SPL-capabilities have a too elaborated *Action* description. It is not simple to shorten such description without losing the essence of the SPL-capability. Similar is the case for Product Line Architecture. The newly introduced Product Line Architecture (PLA) focus area has three SPL-capabilities regarding product line architecture development. However, these capabilities have a rather different 'Action' description compared to all other capabilities. Product Line Architecture has been implemented in the maturity matrix, since most business functions have direct or indirect input for development of the architecture, such as common and variable product features, core assets planning, roadmapping and product lifecycle management. The PLA capabilities need to be validated if they are optimally implemented in the SPM maturity matrix.

Secondly, the positioning of the SPL-capabilities has not been validated. The original positioning of the SPM-capabilities has been performed through multiple case studies. The positioning of the SPL-capabilities has been performed through comparison and analysis with the SPM-capabilities. We tried to avoid replacing SPM-capabilities, giving them another maturity, since we did not have much evidence to base our SPL-capability maturity on. The 3 of the 68 SPM-capabilities that were replaced happened for the reason that a SPL-capability had to be performed prior to that SPM-capability and thus 'pushes' the latter one into a more greater maturity. However, compared to the positioning of the SPM-capabilities more research was performed to indicate the maturities; and the SPL-capabilities should be validated.

One of the SPL-capabilities that were left out due to the expert evaluation was *Architectural Product Derivation*. However, this capability is implemented at the case company and according to planning it will be a capability for the coming years. Unfortunately, due to the expert evaluation the capability has not been further processed.

The expert evaluation of the SPL-capabilities was performed by three practitioners of whom 2 were from the same organization. This number is lower than we were expecting. The expert evaluation gives reliability of the essence to the SPL-capabilities. However, the greater the number of experts are the greater the reliability. In addition, more information was wanted from the situational factors of the expert Companies. The Situational factors regarding the Market and Customer characteristics would have given a better understanding of the context of the expert companies. This would have also benefited the comparison with the case company regarding similarities and differences. The stronger the similarities between the case company and the expert companies, the better the expert evaluation also apply for the case company; this implies that SPL-capabilities are more reliable. Differenced indicates a low coherency; this implies that the evaluation did not make the SPL-capabilities more reliable

Applying the SPLM maturity matrix at the case company involved the three departments most involved with SPM. One of the departments, Marketing & Sales, was represented by only one practitioner. This underrepresentation does not help in making the averaged results more 'average' since the other two departments have three practitioners.

The case study at CCV was useful in order to study an organization with product lines and to research SPL implementations for the maturity matrix. However, the SPL development environment and the implemented SPM processes were experienced as emerging. This implies that much maintained

processes of SPM and SPL development were relatively new and some were being experimented with still. This can be seen in the results of the maturity matrices where more matured capabilities are implemented while less matured ones are missing. A case study which was more mature in its processes would have had more substantive input for this research. However, one of the advantages of this case study was the chance to see an organization go through changes and reacting to what works and what not.

Applying the SPM maturity matrix at CCV is yet another validation of this assessment method for SPM processes. The case study of this research contributes to the validation of the SPM Competence model. In addition, the case study enhanced the applicability of the model and matrix for SPL development.

8.3 Future research

In this section, future research triggers raised from the research will be presented. In the previous section we discussed the shortcomings of this research and pointed out the issues that still remain. Some of these shortcomings we present as future research directions.

Firstly, the applicability of the SPLM Maturity Matrix in product line development environment needs future research. The matrix has been applied only at one case company and the SPL implementations needs to be validated thoroughly. It might be possible that the SPL-capabilities do not apply completely or cover all aspects in another SPL development environment. In particular, the Product Line Architecture focus area should get more attention to research if the PLA-capabilities are clear and applicable for architectural purposes at other organizations. In order to become more reliable, the matrix needs to be (additionally) validated at various PSC's with a SPL development environment.

In addition to the first future research direction, the organizational aspects (Organizational Supportive Input, section 6.2) need further research. The Organizational Supportive Input is identified as essential to the overall SPL success. Future research can look at the possibilities of incorporating e.g. organizational capabilities into the SPLM Maturity Matrix or as a supportive concept for the product management processes.

Thirdly, the role of 'Supplier' as an external stakeholder in the SPM Competence model should receive more research attention. In the related literature (of the Competence model) no reference is made to the Supplier role, in general. However, during our case study, CCV shortly identified the absence of the 'Supplier', since they feel the need for improvements in this area. In our literature study, we found that organizations with product lines often have at least one regular supplier indicating this is a function to be considered seriously and its possible influence on the processes; different relations kept with Supplier compared to Customers, (business) Partners or the market.

Finally, research should be performed on Situational Factors that are specific for SPL development and what the impact of these factors is on the processes. During the case study and the literature study, it was noticed that customer, market and product characteristics differ from organizations with SPL and organizations without. It should be clear if this difference has any impact on the SPM processes and to which extent.

8.4 Theoretical implications

In this section we discuss the implications this research has on existing researches, i.e. we present the contributions this research has on existing literature, namely that of SPM and SPL development (see chapter three).

The SPM Competence model and maturity matrix

As mentioned previously, we followed the SPM Competence model and corresponding maturity matrix by Bekkers & Weerd (2010) as the foundation of SPM for this research and how we deal with SPM. Regarding SPL, the authors of the SPMCM do not discuss the subject of SPL explicitly in their studies. There was no distinction made of the organizations that participated in the research and/or evaluation of the SPM Competence model or maturity matrix which develop SPL. Most of the organizations were likely to develop standard software product or was in a transformation from specific to standard software product. As a capability of the *Product lifecycle management* focus area, *Product lines* is defined and has the purpose to maximize reuse of resources and simplify the development of new product. However, when studying SPL from a product management perspective we identify several key practices that influence the SPM processes described by the SPM Competence model. This indicates that the development of SPL needs to be treated differently and its influence is wider than the *Product lifecycle management* focus area (Bekkers et al., 2010).

Concluding, the SPM Competence model and Maturity Matrix miss dedicated content needed that apply better to organizations with SPL. In our research, we explicitly studied research of organizations that develop SPL. We do not state that the SPM Competence model and Maturity Matrix do not apply to organizations that develop SPL. In contrary, based on our research we can state that the SPM Competence model and Maturity Matrix do apply (mostly) to at least one organization developing SPL. However, both the SPM Competence model and Maturity Matrix need further SPL specification in order to fit a SPL development environment properly.

Figure 14 illustrates the SPM Competence model with the enhancements proposed after this research (see previous chapters). We identified the absence of a *Supplier* role which was added as an external stakeholder which SPL organizations tend to have at least one of. *Product line architecture* is an added focus area in Product planning. The product line architecture has been identified of great importance for the product line and should be initiated as soon as the product line is being planned.

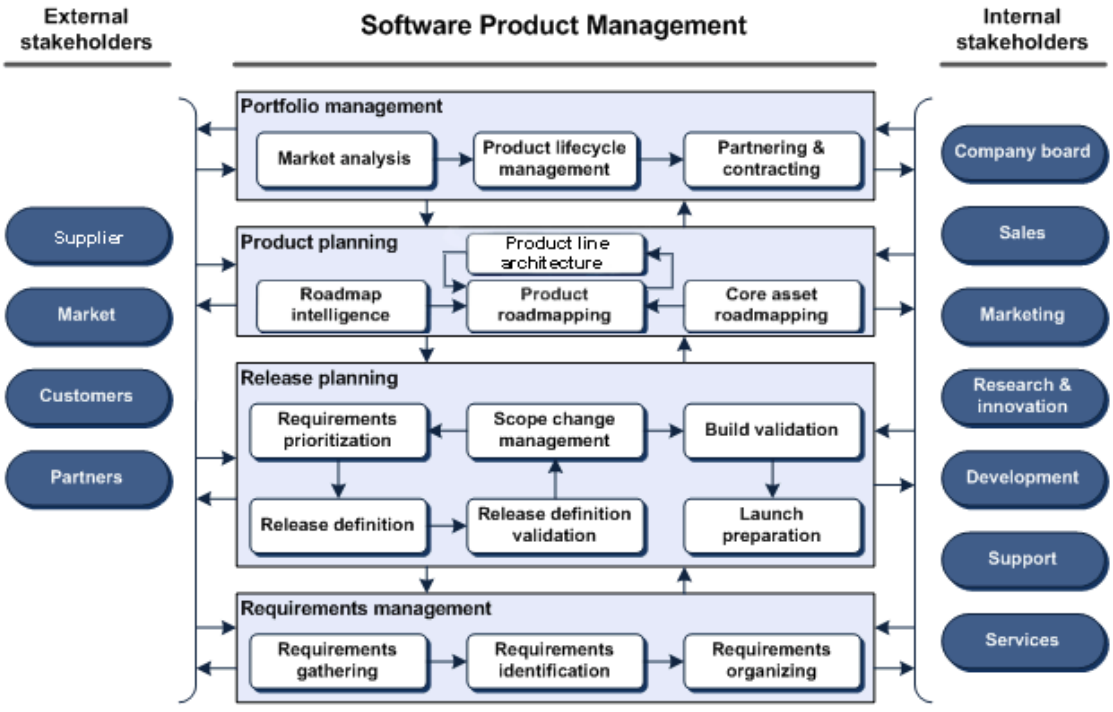

Figure 14: The Software Competence Model with proposed enhancements for product lines.

Bekkers & Weerd (2010, technical report) indicate that not all capabilities are relevant to every type of organization. In studying organizations that develop SPL, we noticed that specific capabilities were missing in the SPM maturity matrix. In addition, our research shows that an organization that explicitly develop SPL indicated that they have a great need for a SPM model which give them structure in their processes and makes improvement possible of already implemented processes; an indication that the SPM Competence model and Maturity Matrix do apply to SPL organizations with respect to the general SPM processes. With respect to this research, no capabilities were indicated as not being applicable for the organization. However, the additions of the SPL-capabilities were experienced as deeper awareness on the SPM processes, i.e. a product line focus. In addition to the SPM maturity matrix, the SPLM maturity matrix (see chapter six) provides capabilities that are product line focused; the enhanced *product line* version of the matrix, with the 16 identified SPL-capabilities. Furthermore, we identified capabilities on an organizational level which are important for the overall success of the product line. These are described in section 6.2 and have not been included in the SPM Competence model nor the SPLM maturity matrix. For the reason that they do not adhere to the principles to be added, i.e. the focus is on supporting the SPL development from an organizational perspective and not product management. Ebert (2007) also describes the importance of organizational structures for SPL organizations, e.g. a product core team.

Software Product Lines

Poor SPM can result in tangible issues such as incorrect content, rework, delays and scope creep (Ebert, 2009). These issues often form a vicious circle: modifications or changes causes unexpected rework which in turn causes delays in the time-to-market which in turn pressures the scope to be reduced, possibly leaving out product features. This is exactly what we noticed at the case company (see sections 7.1 – 7.4). In addition, the root causes (as project symptoms) of these issues identified by Ebert (2009) can be also identified at the case company. However, Ebert (2009) does not explicitly focus on product lines, except the SPM best practices which he presents, can be applied to product lines. The issues identified by Ebert (2009) are recognizable at the case organization. This implies that issues as a result from poor SPM are rather software development wide and not in particular for standard software product or product line, i.e. the development approach.

The distinction starts when looking into the product management processes for a SPL organization. In an earlier study, Ebert (2007) researched SW projects in the telecom industry and in particular embedded systems (similar to the case company). The development approach was product line driven; the challenge was to provide platform products that would have the basic functionalities (common product features, Pohl et al., 1998; Clements, 199) and the customer products which would be tailored (variable product features). Ebert (2007) identified a few ‘best practices’ which positively impact product management, such as product release should be supported by a strong business goal and vision instead of simply collected requirements. Here, requirements, releases, roadmaps, markets are analyzed, discussed and defined for the long(er) term. In addition, product (line) features should be traceable, i.e. planned, communicated, prioritized and monitored especially when involved in core components. On an organizational perspective, Ebert (2007) strongly recommends the introduction of a product core team to enhance stakeholder involvement and commitment. This is important for the overall performance of the SPL.

Although the issues symptoms identified by Ebert (2009) do not distinguish product lines, product lines require specific product management processes when taking the SPM Competence model as our foundation.

Product line architecture

The architecture, often referred to as the reference architecture, of a SPL is of major importance for the development success and achieving the long term advantages (Bosch, 2001, 2002; Böckle et al., 1998; Clements, 1999; Linden, 2002; Northrop, 2002). The input for the architecture is mainly the common and variable product line features (basic functionalities and pre-defined enhanced functionalities) along with quality features. Based on this input, possible framework solutions implementation and development of core assets will be initiated. Furthermore, the authors state that the architecture development should start at an early phase since it is necessary to for product instantiation and will be used as reference for future product line variances. Bosch (2002) presented a matrix relating six different SPL development approaches with SPL artifacts in which the architecture important was in all approaches, i.e. it defines how the product line members are generally decomposed into the core assets. However, none of these authors defined specifically what the needed actions are to gain this input and it was not treated as a product management practice; more development.

The input activities for the SPL architecture are product management related, for these activities ensure the collection of essential information for the creation of the (reference) architecture and decision making. When considering SPL organizations the process of the architecture should be incorporated in product management practices. In this research, we recognized this absence in the SPM Competence model and maturity matrix and therefore defined a new focus area: *Product line architecture*. This focus area has three capabilities, incremental steps, which are meant to be performed to gain information for the architecture and guide the architecture design. In addition, we recommend authors studying SPL from a product management perspective to pay sufficient attention to the processes which are needed as input for the product line architecture. For instance, Northop & Clements (1999) include the SPL architecture throughout their description of Core asset and Product development. They explain the importance of the architecture at each step when it's needed; however they do not elaborate on the activities which provide input needed for the architecture creation.

Domain & Application engineering

Weis et al. (1999) present a SPL engineering approach through a framework which separates to main processes, Domain engineering & Application engineering (DE & AE). Both processes consist of sub-processes which deliver artifacts needed to develop the product line members (see section 3.2). The domain artifacts are reusable and form the platform of the product line and serves as input (reusable artifacts) for the specific applications (or product) development. The application artifacts represent part of the tailored product line applications (product line members) and AE is responsible to manage the product-specific artifacts for each product separately.

Northrop (2002) and Clements (1999) both based their own SPL engineering framework and activities on DE and AE engineering. In their framework, they refer to as Core asset development and Product development. Many other authors (Bayer et al., 1999; Bosch, 2000; Böckle et al., 1998; Linden, 2002; Rombach, 2005; Schmid et al., 2007 and more) who study SPL engineering often refer to these frameworks as the way of executing product line development properly.

However, we experienced otherwise at our case company. At the case company, product line development approach has been introduced these last years and is still being modified to fit the organizations needs best. At the case company, core assets are in very few numbers (not more than 10) and development is not heavily based on the reuse of artifacts. When presented the framework of Weis et al. (1999), the case company stated that it would not be strategically beneficial to focus on the separation of the two processes. Core asset development does take place, however on a rather small scale when compared to the framework of Weis et al. (1999).

It appears that large organizations (e.g. more than 2500 employees such as case company) with large international markets benefit and need a DE & AE clear separation framework due to product

development being more intensively based on (re-)usage of developed core assets to complete the end product (assembling). Weis et al. (1999) use the automotive industry as an example.

For the case company, developing product lines and planning to continue, the much proposed way to execute SPL engineering by existing literature (e.g. Weis et al. (1999) and Northtop (2002)) was too 'product line intensive'. In contrary, the SPLM maturity matrix incorporates the essence of SPL development from a more product management perspective than development. For the case company it was more applicable whilst providing what is needed: structure for the processes that are most important.

References

Akker, J. van de, Brinkkemper, S., Diepen, G., Versendaal, J. (2005). Determination of the next release of a software product: an approach using integer linear programming. *Proceeding of the Eleventh International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'05, 10*, pp. 247–262.

Alves, V., Matos Jr, P., Cole, L., Borba, P., & Ramalho, G. (2005). Extracting and evolving mobile games product lines. In Software Product Lines, pp. 70-81. Springer Berlin Heidelberg.

Ardis, M., Daley, N., Hoffman, D., Siy, H., & Weiss, D. (2000). Software product lines: a case study. *Software-Practice and Experience*, 30(7), 825-47.

Argyris, C., & Schön, D. (1991). Participatory action research and action science compared. *Participatory action research*. WF Whyte.

Baskerville, R. L. (1999). Investigating information systems with action research. *Communications of the AIS*, 2(3es), 4.

Baskerville, R. L., & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11(3), 235-246.

Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., ... & DeBaud, J. M. (1999, May). PuLSE: a methodology to develop software product lines. *Proceedings of the 1999 symposium on Software reusability,* pp. 122-131. ACM.

Bekkers, W., & van de Weed, I. (2010). *SPM maturity matrix*. Utrecht University.

Bekkers, W., Weerd, I. van de, Brinkkemper, S. & Mahieu, A. (2008a). The Influence of Situational Factors in Software Product Management: An Empirical Study, *Proceedings of the 2008 Second International Workshop on Software Product Management*, pp. 41-48.

Bekkers, W., Weerd, I. van de, Brinkkemper, S. & Mahieu, A. (2008b). "Situational Process Improvement in Software Product Management", Thesis report: INF/SCR-08-09. The Netherlands: University Utrecht.

Bekkers, W., Weerd, I. van de, Brinkkemper, S. & Mahieu, A. (2008c). "The Relevance of Situational Factors in Software Product Management", Technical report: UU-CS-2008-016. The Netherlands: University Utrecht.

Bekkers, W., van de Weerd, I., Spruit, M. & Brinkkemper, S. (2010). A Framework for Process Improvement in Software Product Management. In Riel, A., O'Connor, R., Tichkiewitch, S. & Messnarz, R. (Eds.), *Communications in Computer and Information Science: Vol. 99. Systems, Software and Services Process Improvement,* pp. 1-12. Berlin-Heidelberg, Germany: Springer-Verlag.

Berntsson-Svensson, R (2011). Supporting Release Planning of Quality Requirements: the Quality Performance Model. (Doctoral dissertation). Department of Computer Science, Lund University.

Birk, A., Heller, G., John, I., Schmid, K., von der Maßen, T., & Muller, K. (2003). *Product line engineering, the state of the practice*. Software, IEEE,20(6), pp. 52-60.

Bonner, J. M., Ruekert, R. W., & Walker, O. C. (2002). Upper management control of new product development projects and project performance. *Journal of Product Innovation Management*, 19(3), pp. 233-245.

Bosch, J. (1999, May). Product-line architectures in industry: a case study. *Proceedings of the 21st international conference on Software engineering,* pp. 544-554. ACM.

Bosch, J. (2001, July). Software product lines: organizational alternatives. *Proceedings of the 23rd International Conference on Software Engineering*, pp. 91-100. IEEE Computer Society.

Bosch, J. (2002). Maturity and evolution in software product lines: Approaches, artifacts and organization. *In Software Product Lines,* pp. 257-271.

Bosch, J. (2005). Software product families in Nokia. *Software Product Lines* pp. 2-6. Springer Berlin Heidelberg.

Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J. H., & Pohl, K. (2002). Variability issues in software product lines. *Software Product-Family Engineering* pp. 13-21. Springer Berlin Heidelberg.

Böckle, G., Clements, P., McGregor, J. D., Muthig, D., & Schmid, K. (2004). A cost model for software product lines. *Software Product-Family Engineering,* pp. 310-316. Springer Berlin Heidelberg.

Brinkkemper, S., Weerd, I. van de, Saeki, M. & Versendaal, J. (2008). Process improvement in requirements management: A method engineering approach. *Lecture Notes in Computer Science, Volume 5025*, pp. 6-22.

Brown, S. L., & Eisenhardt, K. M. (1995). Product development: past research, present findings, and future directions*. Academy of management review*, 343-378.

Buhrdorf, R., Churchett, D., & Krueger, C. W. (2004). Salion's experience with a reactive software product line approach. In Software Product-Family Engineering (pp. 317-322). Springer Berlin Heidelberg.

Carlshamre, P. (2002b). Release Planning in Market-Driven Software Product Development Provoking and Understanding, *Requirements Engineering, 7*(3), pp. 139-151.

Checkland, P., & Holwell, S. (1997). Information, systems and information systems: making sense of the field.

Chen, L., Ali Babar, M., & Ali, N. (2009, August). Variability management in software product lines: a systematic review. *Proceedings of the 13th International Software Product Line Conference* pp. 81-90. Carnegie Mellon University.

Christiansen, J. K., Hansen, A., Varnes, C. J., & Mikkola, J. H. (2005). Competence strategies in organizing product development. *Creativity and Innovation Management*, 14(4), 384-392.

Clements, P. C., Jones, L. G., Northrop, L. M., & McGregor, J. D. (2005). Project management in a software product line organization. *IEEE Software*, 22(5), pp. 54-62.

Clements, P., & Northrop, L. (1999). A framework for software product line practice. *SEI Interactive*, 2(3).

Clements, P., & Northrop, L. (2002). *Software product lines*. Boston: Addison-Wesley.

Cooper, R., Edgett, S. J., & Kleinschmidt, E. J. (2001). Portfolio management for new product development: Results of an industry practices study. *R&D Management*, 31(4), pp. 361-380.

Cooper, R. & Edgett, S. (2010). Developing a Product Innovation and Technology Strategy for Your Business. *Research-Technology Management, Volume 53*, pp. 33-40(8).

Czarnecki, K., Helsen, S., & Eisenecker, U. (2004). Staged configuration using feature models. *Software Product Lines,* pp. 266-283. Springer Berlin Heidelberg.

Danilovic, M., & Browning, T. R. (2007). Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 25(3), 300-314.

Deelstra, S., Sinnema, M., & Bosch, J. (2004). Experiences in software product families: Problems and issues during product derivation. *Software Product Lines*, pp. 165-182. Springer Berlin Heidelberg.

Deelstra, S., Sinnema, M., & Bosch, J. (2004). A Product Derivation Framework for Software Product Families. *Software Product-Family Engineering*, pp. 473-484. Springer Berlin Heidelberg.

Dul, J. & Hak, T. (2008). *Case Study Methodology in Business Research*. Oxford: Elsevier.

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

Ebert, C. (2009). Software Product Management. *CrossTalk - The Journal of Defense Software Engineering, 22*(1), pp. 15-19.

Ebert, C., & Smouts, M. (2003, May). Tricks and traps of initiating a product line concept in existing products. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 520-525). IEEE Computer Society.

Ernst, H. (2002). Success factors of new product development: a review of the empirical literature. *International Journal of Management Reviews*, 4(1), 1-40.

Fritsch, C., & Hahn, R. (2004). Product line potential analysis. *Software Product Lines,* pp. 228-237. Springer Berlin Heidelberg.

Funk, J. L. (2004). The product life cycle theory and product line management: the case of mobile phones. *Engineering Management, IEEE Transactions* ,51(2), 142-152.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science*, *46*(4), 451-466.

Halmans, G., & Pohl, K. (2003). Communicating the variability of a software-product family to customers. *Software and Systems Modeling*, 2(1), 15-36.

Helferich, A., Schmid, K., & Herzwurm, G. (2006). Product management for software product lines: An unsolved problem? *Communications of the ACM, Volume 49* (12), pp 66-67.

Helferich, A., Herzwurm, G., & Schockert, S. (2005). Qfd-ppp: Product line portfolio planning using quality function deployment. *Software Product Lines,* pp. 162-173.

Helferich, A., Herzwurm, G., Jesse, S., & Mikusz, M. (2007). Software product lines, service-oriented architecture and frameworks: worlds apart or ideal partners? *Trends in Enterprise Application Architecture,* pp. 187-201.

Henfridsson, O., & Lindgren, R. (2007). Action research in new product development. *Information Systems Action Research* pp. 193-216. Springer US.

Jaring, M., & Bosch, J. (2002). Representing variability in software product lines: A case study. *Software Product Lines*, pp. 15-36. Springer Berlin Heidelberg.

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

Krishnan, V., & Ulrich, K. T. (2001). Product development decisions: A review of the literature. *Management science*, 47(1), 1-21.

Krueger, C. W. (2002). Variation management for software production lines. *InSoftware Product Lines* (pp. 37-48). Springer Berlin Heidelberg.

Larson, E. W., & Gobeli, D. H. (1988). Organizing for product development projects. *Journal of Product Innovation Management*, 5(3), 180-190.

Lehtola, L., Kauppinen, M., & Vähäniitty, J. (2007). Strengthening the link between business decisions and RE: Long-term product planning in software product companies. *Proceedings - 15th IEEE International Requirements Engineering Conference, 4384178*, pp. 153-162.

Lindkvist, L., Soderlund, J., & Tell, F. (1998). Managing product development projects: on the significance of fountains and deadlines. *Organization studies*,19(6), 931-951.

McGregor, J. D., Northrop, L. M., Jarrad, S., & Pohl, K. (2002). Initiating software product lines. *IEEE Software*, 19(4), 24-27.

Metzger, A., & Pohl, K. (2007, May). Variability management in software product line engineering. *Companion to the proceedings of the 29th International Conference on Software Engineering* pp. 186-187. IEEE Computer Society.

Muthig, D., & Atkinson, C. (2002). Model-driven product line architectures. *Software product lines,* pp. 110-129. Springer Berlin Heidelberg.

Nebut, C., Fleurey, F., Le Traon, Y., & Jézéquel, J. M. (2004). A requirement-based approach to test product families. *Software Product-Family Engineering*, pp. 198-210. Springer Berlin Heidelberg.

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

Phaal, R., Farrukh, C., Mitchell, R., & Probert, D. (2003). Starting up roadmapping fast. *Research-Technology Management, 46* (2), pp. 27-59.

Phaal, R., Farrukh, C., Mitchel,l R., & Probert, D. (2004). Technology roadmapping - A planning framework for evolution and revolution. *Technological Forecasting and Social Change, 71*(1-2), pp. 5-26.

Pohl, K., Böckle, G., & Linden. F. J. v. (2005). *Software Product Line Engineering: Foundations, Principles and Techniques*. New York: Springer.

Robertson, S., & Robertson, J. (1999). *Mastering the requirements process*. Harlow, UK: AddisonWesley.

Rombach, D. (2006). Integrated software process and product lines. *Unifying the Software Process Spectrum,* pp. 83-90. Springer Berlin Heidelberg.

Schmid, K., & Eichelberger, H. (2008). A requirements-based taxonomy of software product line evolution. *Electronic Communications of the EASST*, 8.

Svahnberg, M., & Bosch, J. (2000). Issues concerning variability in software product lines. *Software Architectures for Product Families* pp. 146-157. Springer Berlin Heidelberg.

Svensson, R. B., Sprockel, Y., Regnell, B., & Brinkkemper, S. (2012). Setting quality targets for coming releases with QUPER: an industrial case study. *Requirements Engineering*, 17(4), 283-298.

Sinnema, M., Deelstra, S., Nijhuis, J., & Bosch, J. (2004). Covamof: A framework for modeling variability in software product families. *Software Product Lines* pp. 197-213. Springer Berlin Heidelberg.

Sochos, P., Philippow, I., & Riebisch, M. (2004). Feature-oriented development of software product lines: mapping feature models to the architecture. *Object-Oriented and Internet-Based Technologies,* pp. 138-152. Springer Berlin Heidelberg.

Taborda, L. J. (2004). Generalized release planning for product line architectures. *Software Product Lines* (pp. 238-254). Springer Berlin Heidelberg.

Taborda, L. J. (2004). Planning and managing product line evolution. *Software Product-Family Engineering,* pp. 296-309. Springer Berlin Heidelberg.

Tatikonda, M. V. (1999). An empirical study of platform and derivative product development projects. *Journal of Product Innovation Management*, 16(1), 3-26.

Tatikonda, M. V., & Rosenthal, S. R. (2000). Successful execution of product development projects: Balancing firmness and flexibility in the innovation process. *Journal of Operations Management*, 18(4), 401-425.

Van der Linden, F. (2002). Software product families in Europe: the Esaps & Cafe projects. *Software, IEEE*, 19(4), 41-49.

Van Ommering, R., & Bosch, J. (2002). Widening the scope of software product lines—from variation to composition. *Software product lines*, pp. 328-347. Springer Berlin Heidelberg.

van Steenbergen, M., Bos, R., Brinkkemper, S., van de Weerd, I., & Bekkers, W. (2010). The design of focus area maturity models. *Global Perspectives on Design Science Research*, pp. 317-332. Springer Berlin Heidelberg.

Vähäniitty, J., Lassenius, C., Rautiainen, K. (2002). An approach to product roadmapping in small software product businesses. *Quality Connection - 7th European Conference on Software Quality (ECSQ2),* pp. 12-13.

Vähäniitty, J. (2004). Product Portfolio Management in Small Software Product Businesses - a Tentative Research Agenda. *Proceedings of the 6th International Workshop on Economic-Driven Software Engineering Research (EDSER-6)*.

Vähäniitty, J., & Rautiainen, K. (2005). Towards an Approach for Development Portfolio Management in Small Product-Oriented Software Companies. *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38)*, pp. 25-28.

Vlaanderen, K., Brinkkemper, S., Cheng, T., & Jansen, S. (2009). *Case Study Report: Agile Product Management at Planon*. Technical Report UUCS-2009-005, Department of Information and Computing Science, Utrecht University.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58-70.

von der Maßen, T., & Lichter, H. (2004). Requiline: A requirements engineering tool for software product lines. *Software Product-Family Engineering* pp. 168-180. Springer Berlin Heidelberg.
Van Der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., & Obbink, H. (2004). Software product family evaluation. *Software Product Lines* pp. 110-129. Springer Berlin Heidelberg.

Weerd, I. van de (2009). Advancing in Software Product Management: An Incremental Method Engineering Approach. (Doctoral dissertation). *SIKS Disseration Series (2009-34).*

Weerd, I. v., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006a). On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. *Proceedings of the 1st International Workshop on Product Management*, Minneapolis/St. Paul, Minnesota, USA, pp. 312-315

Weerd, I. v., Versendaal, J., & Brinkkemper, S. (2006b). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. *Twelfth Working Conference on Requirements Engineering: Foundation for Software Quality*, Luxembourg, pp. 97-112.

Weerd, I. van de & Brinkkemper, S. (2007). Meta-modeling for situational analysis and design methods. *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, Idea Group Publishing, USA: Hershey.

Weerd, I. van d & Brinkkemper, S. (2010). *Lecture notes on Software Product Management*. Utrecht University.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer.

Yin, R. (2003). *Case study research: Design and methods (Vol. 3rd)*. Beverly Hills, California, United States of America: Sage Publishing.

Zhang, H., & Jarzabek, S. (2004). XVCL: a mechanism for handling variants in software product lines. *Science of Computer Programming*, 53(3), 381-407

Appendix A: Complete SPL-practices mapping

SPL-practice	ID	SPL-Activity	SPMCM-mapping	Status	Rationale
Domain Engineering (DE) & Application Engineering (AE)	3	Creation of roadmap for (common and variable) product features	Product planning: PR		Improvement: creation of product roadmap based on common and variable product features of the intended SPL-members. No details are given on the timespan of the roadmap except for "as far as foreseeable". This Roadmap creation is similar with the focus area Product Planning, except for the product features and timespan information.
	4	Product management (PM) defines the release planning (dates) of the SPL-members or specific product or application, based on common & variable features, through the roadmap.	Release Planning: LP		Improvement: Roadmap includes release dates and SPL-members features. Product management is responsible for defining the release schedules of the various SPL-members (marketable products) which is presented in the roadmap with their features. This can be for market introduction or for specific customers. The roadmap is defines the scope of the SPL through the features.
	5	PM deals directly and firstly with Requirements engineering (RE) (first on domain level afterwards on product or application level)	Requirements management	O	Neutral: mostly stating Requirements Gathering and Organizing aspects and describe the process on an organizational level. PM defines the common and variable features of the SPL and the members and includes these in the roadmap, which serves as the scope for DRE. Afterwards, PM defines which products should be derived in ARE.
	6	DE provides reusable artifacts to form the first “bricks” for the platform	-	O	Neutral: DE is responsible for providing reusable artifacts such as requirements specification (textual and modeled), architecture, variability model, SW components, tests, and more which are the essence of the platform that AE will need to perform.
	7	PM defines common and variable features of the SPL-members	Requirements management: RG		Improvement: Requirements management has to consider common and variable product features, when defining them for a SPL. This means that requirements can be organized in these two categories. Common = present in all products, variable = present in individual products. Essential are the two sub-processes, DRE and ARE, of which each is responsible for further specification of the requirements focused on reusability (DRE) and on variability of the domain requirements (ARE).
	8	RE differentiate between common and variable features	Requirements management: RG		Improvement: RM should differentiate between common and variable requirements when identifying product features. Identify more common than variable requirements, as variable requirements assure more complexity. However variable requirements are necessary for the essence of the SPL, which is the variability each product or application will have.
	9	DRE provides (reusable) requirements specification for the	Requirements management	O	Improvement: Requirements management for DE is focused on delivering reusable requirements artifacts. This indicates that one

		next sub-processes: Design, Realization and Testing			process has to be focused on providing reusable assets (DE) and another process in exploiting these reusable assets for producing products or applications (AE).
	10	PM defines the products or applications that will be derived from the SPL by defining the different features of the particular products or applications	Requirements management		Improvement: By defining the difference of the features (variable requirements), the different SPL-members are recognized. Product management defines which product or application should be derived in ARE by prescribing the features of the product, i.e. which product should have which common and variable features. Specific customer-requirements are added in this sub-process.
	11	ARE provides requirements artifacts for specific product or application	Requirements management	O	Improvement: Requirements management for AE is focused on requirement artifacts for individual products or applications.
	12	DRE and ARE communicates back to PM for additional or altering features	Requirements management	O	Requirements management for DE and AE report back to Product management if requirements (common, reusable or specific to a product) need to be altered or added; forming a feedback-loop.
	13	Variability management is dealt with in RE	Requirements management: RO		Improvement: RM should manage the variability, which is identifying, documenting and modeling the variable requirements. RM explicitly document and model variability (external variability= visible to customers, possible to choose variants). This entails variable requirements and modeling (variation points, variants and their relationships) of this variability. Here the variability diagram is created and presents the differences between the members.
	14	Architecture design is driven by RE	Requirements Management	A	Improvement: RM should implement some practice towards architectural design. For instance, quality requirements should count for the architectural design or grouping of requirements according to architectural concerns. Common and variable requirements and the variability model are passed onto Domain Design which translates the requirements to technical solutions in the SPL architecture. Especially quality requirements (performance, security, usability, etc) are the drivers for architectural design. The Variation/variability in requirements often results in variation/variability in the architecture. Component frameworks are used to support the various types of quality requirements, i.e. frameworks are used to model SPL requirements in a structured manner into components with their relationships. It also incorporates, properly, quality requirements such as flexibility, maintainability, evolvability.
Core asset, Product development and Management	16	(Technical) Management monitors the processes of Core asset (DE) and product development (AE)	Overall	O	Improvement: Get (top) management more involved in the DE & AE processes, e.g. by monthly reports or management involved, e.g. as a stakeholder for instance at the end of each Sprint when following agile Scrum. However, it is relevant for the overall success of SPM processes.
Variability	18	Variability in SPL is determined by	Requirements management: RI, RO		Improvement: Determine which requirements will differentiate between

management		the variable features between the SPL-members			the SPL-members. Focus area's RI for identifying the variation in the requirements, RO for organizing the common and variable features and the modeling there of. The identified variability has to be modeled by a variability-modeling technique, e.g. Feature Modeling technique.
	19	Requirements traceability is needed	Requirements management: RO:B		Improvement: Register in which (core) asset or component a requirement will be implemented as extra requirement data. The capability of RO:B logs requirements' data expect for in which asset a requirements will be implemented.
	20	Variability management is part of RM	Requirements management		Improvement: Variability management is an activity that occurs in Requirements management, i.e. when identifying common and variable features.
	22	Reusable Components represent a set of functionalities of the products	Requirements management		Improvement: Requirements can be organized according to components explicitly. This statement states a set of functionalities or requirements form a component.
	23	Components can be reused for a number of products or applications	Product planning: CAR		Neutral: Components are assets that are used for building SPL-members. In SPL engineering component-development needs to be planned, i.e. through the roadmap.
	24	A component in the architecture implements a particular/coherent domain or set of functionalities (requirements)	Requirements management: RO	A	Neutral: A component consists of functionalities that are closely related or together form a solution. Similar to the Requirements organizing focus area.
	25	Components are stored in a component repository for product development	-		Neutral: Components (core assets) that product development will need to develop the end products are stored in a component repository.
	26	SPL architecture is a generic architecture consisting of components, connectors and additional constraints	-	A	Neutral: The SPL-architecture is generic for all SPL-members, referred to as reference architecture. The reference architecture consists of components, connectors and constraints.
	27	Role of the architecture is to describe the commonalities and variabilities of the products in the SPL, and to provide the overall structure.	Requirement management	A	Improvement: The commonalities and variabilities should be identified and organized in requirements management. In addition, requirements management is direct input for the architectural design
	28	SPL-members are instantiations of the reference architecture and components in the architecture	-	A	Neutral: The reference architecture is used to derive the architecture for each SPL-member. The product architecture presents the components necessary to build the product.
	29	Features of the SPL products should be predicted in a time span of 5 years in the future.	Product roadmapping		In the roadmap the vision of product development is presented for the coming years. For SPL this is also detailed into features that will be expected in the next 5 years in the SPL
	30	Variability occurs in different levels	-		Variability in the design phase can have impact on different levels.

		in the design			
	31	Product line level			Variations between SPL-members, different components
	32	Product level			Architecture and components selection for a particular product
	33	Component level			New implementations of interfaces and evolution thereof
	34	Sub-component level			Features selection to create a component
	35	Code level			Where variability actually takes place, is implemented
	37	SPL architecture has to support planned changes	-	A	Neutral: The SPL-architecture has to support the planned changes, which is variability
	38	SPL architecture records components dependencies	-	A	Neutral: The SPL-architecture records the components' dependencies in order take into consideration during development.
	39	Properly adapting architecture	-	A	Neutral: Properly adapting the architecture demands proper documentation, methods, techniques and guidelines for handling variability with all stakeholders customers, development, management, etc.
	40	Product derivation from an architectural point of view	-	A	Neutral: The architecture supports all features, HW and SW, and disabling certain features create the particular products (maximalist approach)
	41	Identifying variability is often based on analyzing commonalities and differences between SPL-members	Requirements management: RI, RO		Improvement: Commonalities and differences identification and analysis should be part of RM, since it comes down to common or different features and requirements
	43	Variability is generally expressed through variation points.	-		Neutral: Variation points and variants are a technique to model and represent variability and the variations, referred to as Feature Graph Modeling. A variation point refers to a delayed design decision, i.e. it indicates a specific point in development or deployment phase of a SW system.
	45	Differences between SPL-members are well documented	-		Neutral: The differences between SPL-members are exclusively documented, this is referred to as variability. These differences can be modeled in variation points and variants in a Feature model.
	46	A feature is a logical unit of behavior that is specified by a set of functional and quality requirements.	Requirements management		Improvement: The identification of features, functional and quality requirements are part of requirements management; requirements can be organized according to features and in turn, features into components. Features are abstract from requirements. A component has features and a feature has requirements
	49	Product instantiation in SPL	-		Neutral: Change before product instantiation (anticipated) relies on the reuse infrastructure (DE), whereas change after instantiation(unanticipated) relies on non-reusable code(AE)
	50	Optimal phases to introduce and bind variation points	-		Neutral: Selecting the optimal phases to introduce and bind variation points in the SW life cycle has considerable impact on flexibility (variability rate) of the system as well as the development and

					maintenance costs. Bind a variation point too early, flexibility is less than required. Bind a variation point too early, flexibility is less than required
RequiLine	54	RM differentiate between common and variable requirements	Requirements management: RI		Improvement: RM should differentiate between requirements for all SPL-members (common) and requirements for specific SPL-member (variable)
	55	Requirements/feature traceability is needed	Requirements management: RO:B		Improvement: Features should be linked to the SPL-member they are implemented in. This makes it possible to trace requirements down (history of requirements implementation) and have the knowledge of how an issue has been solved before.
Generalized Release planning for SPL	57	Requirements allocation and traceability	Requirements management: RO:A, B		Improvement: Bundle requirements that fit together and register where this bundle will be implemented, organized per SPL-member inclusion. Both components and end-product are linked in the Release matrix. This implies requirements being allocated to components and finally the end-products. Traceability too, requirements can easily be traced in which product they were implemented
	58	Release planning for components	Release planning: RD:B, C		Improvement: Plan the release of components and products in the Release matrix, as is needed to meet the release date (the component producer is the one responsible for requirements prioritization).
	59	Requirements management differentiate between shared and variable components	Requirements management		Improvement: Requirements management categorizes requirements in shared domain components (common) and product-specific components (variable)
PPP-QFD	61	Requirements identification AND prioritization by customers	Requirements management: RG:E – Release planning: RP:C		Similarity: Requirements for the SPL are firstly gathered from existing and potential customers. These requirements are analyzed and sorted. Secondly, the existing and potential customers are asked to prioritize the requirements.
	62	RM identifies Product line members	Requirements management: RO – Release planning: RP		Improvement: Requirements can be sorted and organized in such a way, that segments can be extracted. Based on the prioritized requirements, customer segments are derived using cluster analysis. Each product line member is identified using the rule 'one product line member per customer segment'. Experts provide input on a technical level.
PuLSE	66	Management path SPL future	Portfolio management		Neutral: The product line scope is initiated by business objectives defined by the stakeholders. Management defines the business objectives that drive the SPL initiation.
	67	Product map defines product line scope	Product planning: RI		Improvement: to present requirements with SPL-members together with other valuable information for the product line, i.e. costs, benefits, market, objectives and competitors in a product map. A matrix with the SPL-members in the columns and the characteristics in the rows together with information on the market, costs, competitors and benefits.
	68	Modeling input for Architecture	-	A	Neutral: Information on the product line is elicited and modeled in

					different views such as workflow diagrams, sequence charts and data models. The Decision model, where SPL-member specification can be derived, contains a structured set of decisions of which each corresponds with a variability in concepts, requirements, features, etc. of the SPL. The Decision Model makes it possible to derive product variants and is similar to the Variability Model/diagram mentioned in other diagrams.
	69	Architecture is driven by requirements	Requirements management	A	Improvement: common, variable and product-specific requirements are used to initiate the architectural design of the SPL, i.e. concepts are determined and modeled according to their relation and product line scope. The aim is to define a domain-specific SW architecture that covers the existing products and future SPL-members
	70	Requirements are also used for product validation	Release planning: RBV		Similarity: validation of the end product is performed to assure product quality according to the requirements set beforehand.
	71	Design and coding is validated against the architecture	Release planning		Improvement: the reference architecture is used for the validation of the design models and SW-coding to validate if the limitations and structure of the architecture are met.
Product Derivation Framework	73	Requirements traceability is essential	Requirements management: RO:A, B		Improvement: requirements are also linked to the core asset in which they've been implemented. Requirements are organized based on shared core assets is a similarity.
	74	Product configuration is validated against the requirements	Release planning: RBV		Similarity: validation of the end-product, to assure product quality, according to the requirements set beforehand.
	75	Requirements drive Product Derivation	Requirements management		Neutral: SPL-products are mostly configured or assembled from existing SPL-assets and some tailoring. Based on the requirements management provides, the necessary assets can be chosen for the initial configuration. RM has the responsibility to update requirements when (and during) these are chosen for configuration or included in pre-implementation processes (e.g. when customers wishes changes). Usually, the initial configuration is iterated until it is proper (re-architecture, re-components and re-parameters) and the end-product is declared ready. Requirements that are not known by RM, i.e. represented by core assets, and have to be included in the end-product can only be included through adaption in architecture and components if this cannot be tailored in the end-product (depending on the impact of the requirement).
	76	Product architecture is derived from the reference architecture	Architecture	A	Neutral: The product architecture is derived from the product line reference architecture, for as much as possible.
	77	Product roadmapping	Product planning: RI, PR		Similarity: the domain and scope of the SPL as well as its future developments (evolution) are predicted in combination with a technology scope in a roadmap. However, no timespan is given.

Integrated SPPL	79	SPL engineering approach reduces TTM projects	-	O	Neutral: This is an interesting statement for the case company.
	80	SPL engineering exists of 2 separate development processes: DE and AE	-	O	Neutral: One process, DE, is responsible for developing reusable components and set up the product line development platform. The other process, AE, uses mainly the reusable components developed by DE to build the end-products, and tailors where this is needed.
	81	SPL engineering promotes proactive reuse of pre-designed commonalities and controlled variabilities within a family of systems	Requirements management: RG		Improvement: pre-designed commonalities and controlled variabilities are common and variable features between the SPL-members. These should be handled in RM
	82	Commonalities and variabilities are implemented through a components architecture	Requirements management	A	Neutral: common and variable features are organized into components. The architecture describes how components should be implemented and how they relate to each other.
	83	Reusable artifacts (DE), are kept in artifact repository			Neutral: The reusable artifacts, from requirements to test cases, that is needed during product development are stored in an artifact repository.
	84	(predefined) Variability choices (variants) are linked to corresponding components	Requirements management: RO		Improvement: Features (common or variable for SPL-members) are linked to the corresponding component that should be implemented in order to realize the features.
	85	Requirements engineering process is fundamental in DE process in order to achieve SPL advantages	Requirements management	O	Improvement: Requirements management should receive proper attention and should not be underestimated during SPL engineering. The highest focus lies with handling (identification, organizing, analysis, modeling, documentation) commonalities and variabilities.
SPL FAST Process	87	The FAST process exists of two phases: DE and AE	-	O	Neutral: DE and AE are recognized as the two main processes essential for SPL engineering. Similar to other SPL-practices.
	88	FAST process	-	O	Neutral: FAST is applicable as a SW development process when organizations create multiple versions of a product with significant common attributes: behavior, interfaces, code.
	89	A process called Commonality Analysis	Requirements management		Improvement: The common and variable characteristics (features) of a product family are identified and analyzed and documented in a primarily natural language document (text) document.
	90	Commonality analysis document as communication tool	Release planning: RD		Similarity: The commonality analysis document is a powerful communication tool between Marketing, senior Management and developers.
	91	During commonality analysis example scenarios are used to explore differences between SPL-members	Requirements management: RI		Improvement: Use these example scenarios (techniques) in order to further analyze common and variable features when this is needed for certain products.
	92	Usability scenarios	Requirements management: RI		Improvement: Describes actions required to perform common user operations

	93	Variability scenarios	Requirements management: RI		Improvement: Emphasize the differences between individual products
	94	Prototyping a SPL-member makes it possible explore (deeper) differences between the members by using scenarios	Release planning		Improvement: Prototyping allows for deeper identification and analysis of variable features and other aspects between SPL-member when scenarios are not sufficient.
	95	Design patters in SPL	-		Neutral: Design patterns in SPL-engineering mostly focus on variability. This in turn is important for the product line architecture.
	96	Binding in SPL	-		Neutral: Binding times and values of the variability are essential and should be done carefully, since the variability depends on this.
	98	SPL development requires more effort than single product development	-		Neutral: This is an interesting statement to show the difference between single product development and product line development
BAPO	100	RM is input for the architecture design	Requirements management		Improvement: The design for the reference architecture receives input from RM, i.e. the functional and quality requirements form the design of the reference architecture. Both commonalities & variability (variation points and variants) should be modeled in the architecture. The reference architecture defines the components (mandatory, optional, and alternative), component interrelationships, constraints, and guidelines for use and evolution in building systems of the SPL.
	101	Traceability of requirements is vital in RM	Requirements management: RO:B		Improvement: RM should trace requirements to know in which assets they are implemented, for maintenance reasons and a complete manageable process. Traceability is connected with configuration and version management for the configurations and version of the particular assets and components.
BAPO evaluation	104	Business: Identity	Product planning: CAR	O	Neutral: Existing family assets are reused in product development for opportunistic reasons. Likewise, make/buy/mine/commission SPL assets are only done for opportunistic reasons. This is information that is not to be missed. However, it is not more than that.
	105	Business: Vision	-	O	Neutral: Product line scoping is based on expected future products and which product will not be produced. The marketing of these future products are also discussed. Roadmapping plans the development of the SPL and decides on make/buy/mine/ commission assets.
	106	Business: Strategic planning	-overall	O	Neutral: The strategic planning of the SPL includes SPM activities such as SPL requirements definition. The product portfolio is evaluated and cheap products are pushed while expensive ones are dropped. Furthermore, product line scoping is based on the expected ROI of the entire SPL. The roadmaps are based on intra-company agreements, TTM and profit estimations. The SPL is marketed as a whole and the marketing is aligned with product development.

	108	Architecture: Reuse level	Product planning	A	Neutral: asset sharing is only beneficial when the commonalities are clear to be exploited. Domain-specific components can be acquired from external sources if this is more beneficial (less effort) than building them.
	109	Architecture: Product quality	-	O	Neutral: intra-organizational reuse of assets takes place through a platform which provides domain functionality that is applicable for all products, i.e. commonality. Non-commonalities are implemented in individual application or product (product derivation)
	110	Architecture: Product family architecture	-	A	Neutral: Increase of general functionalities that are applicable for most products are introduced through the product line platform. This should be supported by the architecture. Specific functionalities that are present for one or few products takes place through product derivation.

Appendix B: SPL-Capabilities (pre-evaluation)

Requirements management

Requirements gathering

Title	Basic product line scoping (7, 8, 10, 47, 48, 54, 59, 64)
Goal	Define product line features to support the scope.
Action	Requirements management defines *Common*, *Variable* and *Product-specific* product features. A Common feature is present in all or most products. A Variable feature is present in some products only. A product-specific feature is present in only one individual product (customer wish). A product feature is a logical unit of behavior that is specified by a set of functional and quality requirements, implying a feature is defined by multiple requirements. By defining the different features, and thus requirements, and in which product they will be present (variable features), the different product line-members can be identified. The aim is to define more common than variable requirements, as variable requirements assure more complexity. However variable requirements are necessary for the overall variability of the SPL.

Requirements identification

Title	Advanced product line scoping (91,92,93,94)
Goal	In-depth analysis of product features.
Action	After commonalities and variabilities have been identified, example scenarios/real-life scenarios are deployed for further analysis and evaluation of the commonalities and differences. Usability scenarios describe actions required to perform common user operations. Variability scenarios emphasize the differences between individual products. In some cases, differences are difficult to identify and/or evaluate. Prototyping a product line-member makes it possible to explore profound differences between the members. Prototyping is vital in cases where a product line is developed, specifically for a niche market or customer.

Requirements organizing

Title	Product line features organizing (22,23,24)
Goal	Organize features according to (reusable) components
Action	Organize features together that serve the same purpose or functionality to form components, i.e. features that complete a function of a component are grouped together. A (Reusable) component represents a set of closely related functionalities/features that form a product solution.

Title	Components dependency registration (38)
Goal	Record the dependency of the components
Action	Determine and register components' dependencies. Components can depend on other components in order to function properly or conflicts may occur. These dependencies are direct input for the architecture. Components' dependencies should be described when the corresponding features are organized accordingly. This can be done textually and/or modeled (the latter one often communicates easier).

Title	Product line requirement life cycle management (55,57,84,101)
Goal	Make requirements traceable - Requirements can easily be traced in which product(s) they were implemented.
Action	Register in which product line-member the components, and thus features, are implemented in. This is necessary in order to be able to trace requirements down (history of requirements implementation) and have the knowledge of how an issue has been solved before and can be done again (reuse) and for an overall manageable process. This start with requirements constituting features, features being bundled into components and components being developed

	and (re)used to build the end-product; the traceability of the requirements during the whole trajectory is important, e.g. it should be clear which SPL-members should get updated versions of core components when these are available.

Title	Variable feature management (13,18,41,42,43,44,45)
Goal	Manage variable product line features properly.
Action	Variability (variable features) is identified and explicitly modeled and documented. This is referred to as Variability management. Identifying variability is often based on analyzing commonalities and differences between SPL-members; especially external variability which is visible to end-users and possible for them to choose variants. This happens while gathering, identifying and defining product features. The variable features are destined to be implemented in (only) some individual SPL-members, unlike the common features, which are implemented in all members. The variable features and the variances are modeled into a Variability diagram with Variation points (decision points), variants (a decision) and the relation thereof with proper textual documentation. This is also known as Feature Modeling technique. The purpose is to have a clear view of the variability of the product line and to be able to manage it, since this is vital for the product line success.

Release planning

Requirements prioritization

Title	Components consideration (58)
Goal	Product line features prioritization
Action	Prioritize the features that will be implemented in end-product from the next release on by assigning priority to them (prioritization techniques can be used). This prioritization is performed with the end-product(s) in focus and which component is required to complete the product. This implies that features would be implemented in components and these components would complete the end-product. Prioritization is necessary, since not all features can be implemented, due to costs, resources and market introduction deadlines. Hence, the features with the desired priority will be included in the particular components.

Release definition

Title	Product line release definition (4,58)
Goal	A selection of features for implementation based on priority
Action	A practical selection of the features is made given the limitations on engineering resources, based on the priority assigned. The function and essence of the components is also considered when making the selection. The selection is defined textually which will be necessary for further steps.

Title	Product line release planning (4,58)
Goal	Release plan for the product line members
Action	After the prioritized features are formed into components (sharing same functionality, same core asset, same member, etc.), the release can be planned. A release plan is created based on the product line roadmap, i.e. release dates are determined for the product line members (end-products), supported by detailed information on the components, common and variable features they are composed of, from the next release on. This can be for market introduction and for specific customers. The rationale behind this is that components can be reused for future releases and products. This is part of maintaining the development platform for the product line.

Release build validation

Title	Architectural release validation (71)
Goal	Release validation by architecture – Release quality assurance
Action	The design and coding of the SPL-members (the build) is validated through the product line architecture before the actual release is launched. The design and software code have to adhere to the limitations and structure of the product line architecture. This validation is performed by the department(s) who is (are) responsible for developing and maintaining the product line's

	architecture(s). The product line's architecture is vital for achieving the business goals set up front, when management decides to engage in a software product line development approach. Hence, the necessity for validation through architecture.

Product planning

Product roadmapping

Title	Product line roadmapping (3,29,46,105)
Goal	Define the scope of the software product line through a roadmap.
Action	A roadmap is created detailing the anticipated products of the product line, its members, as far as foreseeable. The members are represented by the components (if possible) of which they are built off, i.e. multiple features form a component, whereas a feature abstract from requirements. The product line features, common and variable, should be predicted for a time span of 5 years. However, other authors (e.g. Svanhberg et al.) believe this is not practical, since a great amount of the future requirements (not technology shifts and/or other development changes) of the product line cannot be predicted.

Core asset roadmapping

Title	Core asset usage (3,29,46,105)
Goal	Intra-organizational reuse of core assets – exploiting core assets.
Action	Increasing commonalities will need to be managed in order to be exploited properly. The managed commonalities are developed into fundamental components (core assets). These core assets, which mostly contain domain functionality, amongst other shared assets, are reused by other (internal) departments through the product line platform. Features that are shared by sufficient members are included in the core assets, whereas features shared by only few members are developed as part as product derivation. This ensures knowledge sharing and more efficient product development as more reuse is taking place. This is a typical of product lines practice.

Roadmap intelligence

Title	Visualization of the product line scope (67)
Goal	Visualize product line scope information to clearly communicate with stakeholders, e.g. other departments, customers, suppliers.
Action	The members and their features are represented in a product map, i.e. a matrix with the members in the columns and the features in the rows together with information on the market, costs, competitors and benefit (not date nor schedule is included, not to be mistaken it with product roadmap). This is also a way to represent traceability; to allocate features to members together with other valuable information for the product line. The above mentioned example is a way to visualize the product line scope. It is possible that software organizations use other ways to visualize the same type of information.

Portfolio management

Product lifecycle management

Title	Financial Product line scoping (106)
Goal	Scope the product line with information on costs and profits.
Action	The decision whether a product will or will not be part the product line scope is based on the expectations of the ROI. If these are beneficial for the organization, the product will be part of the line, otherwise it will be declined. This can also be applied to features, i.e. features are dropped according to their expected added value or revenue.

Title	Architectural product derivation (40)
Goal	Derive products from the product line reference architecture (maximalist approach).
Action	The reference architecture implements all features (hardware & software). Product derivation from an architectural point of view implies disabling certain features, minimalizing the amount of features, and creating the particular SPL-members with those selected features (Generally, the

	reference architecture is used as basis to further specify as needed for each SPL-member).

Not coupled to any business functions

Product line architecture

Title	Reference architecture construction(14,24,27)
Goal	Create product line reference architecture.
Action	Create the product line reference architecture. Role of the architecture is to describe the commonalities and variabilities of the products in the product line and to provide the overall structure. Common and variable features, represented by the components, are the main drivers for architectural design together with quality requirements such as performance, security, usability, etc. These are also represented in the architecture. Component frameworks are used to support the various types of quality requirements, i.e. frameworks are used to model features in a structured manner into components with their relationships. The reference architecture has to solve issues of variability and reusability. In addition it also properly incorporates quality requirements such as flexibility, maintainability, evolvability. When common and variable features are considered in a very early stage then more flexibility is assured for the product line. As features are represented in components, the components in the architecture implement a particular/coherent domain of functionality, e.g. the network communication domain.

Title	Reference architecture construction based on scenarios (69)
Goal	Create product line architecture with requirements input.
Action	The aim is to define a domain-specific software-architecture that covers the existing products and future SPL-members. This is driven by the functional requirements (generic scenarios; which are similar to commonalities) or domain-independent quality aspects (property-related scenarios; which are similar to variable features with a quality focus). The generic scenarios are combined with property-related scenarios and rated according to architectural importance. Iteratively, functional requirements from the generic scenarios are chosen to create the initial architecture. In each iteration, other generic scenarios are added to the candidate architecture to complete it and refine it, which may result in multiple architectures. The property-related scenarios are used to validate and refine the candidate architecture. Requirements (common, variable and product-specific) are used to initiate the architectural design of the software product line, i.e. concepts are determined and modeled according to their relation and product line scope.

Title	Reference architecture construction based on variability(100)
Goal	Product line architecture development driven by requirements management.
Action	The design for the reference architecture for the product line receives input from Requirements Management, i.e. the functional and quality requirements form the design of the reference architecture. The architecture should relate all design decisions made, to the requirements (functional + quality). Both commonalities and variability are modeled in the architecture. Variability in the requirements is modeled through variation points and variants, i.e. where members may vary from each another. The reference architecture defines the components (mandatory, optional, and alternative), component interrelationships, constraints, and guidelines for use and evolution in building systems of the software product line. Later, the reference architecture is used to create an instance and further specify it for a particular (new) member. When designing the architecture, quality predictions of all products have to be taken into account, which makes it a difficult issue. Hence, having the right description and mechanisms to do so, eases the architecture modeling, e.g. variability modeling. The most important architectures issues that have to be addressed properly are: significant architecture requirements (functional requirements & quality requirements that are significant for the SPL architecture), Concepts (the architecture concepts clarifies the architecture organization), Texture (standard solutions for implementation problems) and Structure (internal

Organizational

Title	Domain & Application engineering (6,80,87)
Goal	Create a product line environment
Action	Create two processes: one process, Domain Engineering, that is focused on developing reusable artifacts and a product line environment (textual and modeled requirements specification, architecture, design, variability model, software components, tests plans, and more) which forms the development platform; and one process, Application Engineering, that focus on developing sellable end-products that are built from the artifacts developed in Domain Engineering

Title	Requirements engineering planning (5,7)
Goal	Plan requirements engineering for product line developments.
Action	Product management controls Requirements engineering directly, first on domain level afterwards on product (or application) level. Product Management defines the common and variable features of the product line in the roadmap, which serves as the scope for further requirements engineering tasks such as detail specification and modeling. Afterwards, Product Management defines which products or applications should be derived in Application Requirements Engineering, implying which requirements will be implemented for which product, individually (variable features)

Title	(Upper) Management Involvement (16)
Goal	Proper execution of product line processes.
Action	(Upper) Management is actively involved and monitors the process of Domain Engineering and Application Engineering. This is for guidance on keeping the processes and sub-processes on track and not to drift away from the business goals. Technical management focuses on Domain & Application Engineering, whilst Organizational management focuses on the best fit organizational structure for the organization. (Upper) Management is more involved in the above mentioned processes to keep, e.g. by monthly reports or management involved, as a stakeholder, at the end of each Sprint when following agile Scrum. However, this capability is relevant for the overall success of SPM processes.

Appendix C: Evaluation results of SPL-Capabilities.

Requirements management

SPL-Capability	Basic product line scoping
Usefulness	3/3
Rationales summary	It is important to know if the feature/functionality is variable before development start. Literature suggests that in order to build in the variability properly, it has to be known in an early phase.
Modification	None

SPL-Capability	Advanced product line scoping
Usefulness	1/3
Rationales summary	It is not useful when fewer and larger variabilities are at stake. This means that it is useful for many and smaller variabilities. However, it is useful to detect early hazardous behavior of the system (similar of the focus area Release build validation).
Modification	This capability needs a condition: useful IF variability in the product line is large in number and it is relatively small variation.

SPL-Capability	Product line features organization
Usefulness	1/3
Rationales summary	Hard to know in advanced which components will be core assets, i.e. reused. This is for an organizations where variability information and requirements tends to appear very late in the product development lifecycle (2 experts). Organizing features according to components improves requirements tracing. The dependencies (interdependencies), i.e. links between features might improves
Modification	This capability will be left out. It seems too large of a challenge for organizations to know this in an early phase.

SPL-Capability	Components dependency registration
Usefulness	3/3
Rationales summary	This is useful and gives great benefits when the impact of changes on components is being analyzed. It is very hard to maintain updated; easy for the easy ones and very hard for the hard ones. In order to really benefit from this capability you need to include many product from your product line (per installation variant maybe too).
Modification	This capability needs a condition: the more products this is applied to the greater the benefit will be. This goes out for all products.

SPL-Capability	Product line requirement life cycle management
Usefulness	3/3
Rationales summary	The value of this capability is seen when it is time to test/validate the product. This way it is clear which requirements are performing how. Nevertheless, mapping requirements to SW-code is usually very hard.
Modification	None

SPL-Capability	Variable feature management
Usefulness	2/3
Rationales summary	The visibility this capability will provide is beneficial. However, by dividing common development and tuning or configuring a specific product you want to increase scalability. However, what you end up with is a situation where one practitioner has the domain knowledge (and product knowledge because that person has

	investigated the detail requirements for the product and is responsible for implementation in the common or variable components) then a third person is expected to be able to configure and understand the requirements and configuration language without hardly any domain knowledge. In theory this seems more feasible than in practice, considering it might requires quite heavy information transferring and tool support to make this work smoothly. From a developer perspective it is only relevant that a feature is included in the product or not; and not whether it is variable or not.
Modification	They capability will be left out. Besides being useful, expert find it not practical in real life and too farfetched.

Release planning

SPL-Capability	**Product line features prioritization**
Usefulness	3/3
Rationales summary	It is important to prioritize requirements, since not all can be implemented at once. As long as each requirements/feature have a business value, implementation costs and architectural implication. If the feature or components is reused in many components it should get a higher market value
Modification	None. However this capability is described on a focus area level (Requirements prioritization). The emphasis of this capability needs to be put on components consideration when prioritizing.

SPL-Capability	**Product line release definition**
Usefulness	3/3
Rationales summary	This is the following step after prioritizing the requirements and the focus should go according to importance. Sometimes the focus is too much on business priority and too little on technical benefits and limitations, when making the selection for development.
Modification	None

SPL-Capability	**Product line release planning**
Usefulness	3/3
Rationales summary	Yes, such a plan is beneficial. However, reuse needs to take place where and when it is beneficial (where it makes sense) and not all the time (100%). This capability is also useful for verification or validation of the end-product, since this would be a specific configuration of features and components. Be careful with forcing as much reuse as possible (100%) with the common components. This might form situations where all products will share the risk with each other. E.g. If a low priority feature is delayed or causing instability, this might stop multiple if not all release of the other products in the product line. Also, too much upfront planning lets the resistance of having great amount of differences in the products gets too high, meaning it will be more difficult to have a lot of variabilities between products.
Modification	Information will be added on control of the upfront planning and to apply reuse only when and where it makes sense. Do not apply limitless reuse.

SPL-Capability	**Architectural release validation**
Usefulness	3/3
Rationales summary	Beneficial, however it might be better to do this during development, since if a mistake is detected then it rarely happens that a market release will be delayed because the code is not following internal standards (1 expert). By not following the PL architecture, other SW development activities might be affected, e.g. tests modules might not function properly or certification of the source code will be challenging, because constraints in the architecture might be invalidated.

Modification	None

Product planning

SPL-Capability	**Product line roadmapping**
Usefulness	3/3
Rationales summary	Good to have that kind of plan. It might be useful to predict upcoming products and what type of feature or components they will contain. This can then be used to minimized component dependencies and design for change in the PL-architecture.
Modification	None

SPL-Capability	**Core asset usage**
Usefulness	3/3
Rationales summary	Useful, however it is important to protect and clearly divide development responsibility here. Some 'reuse agreement' which enables flexibility, will need to assure that the team responsible for product development (AE) do not burden the team responsible for core assets (DE) too much. This can lead to a negative effect on the core assets development team by constantly switching of context which will lead to a decrease of productivity. In addition, sharing product development knowledge benefits development in general, i.e. more efficient and effective.
Modification	This capability needs to be split up into two separate capabilities for a smoother implementation. One capability needs to standardize the infrastructure and the necessities for core asset development (reuse). The other capability would extend the first one to allow internal use of the core assets as well as evolution of the infrastructure/platform.

SPL-Capability	**Visualization of the product line scope**
Usefulness	2/3
Rationales summary	Very useful to note that the difference in detail/abstraction level between SW developers and product planners. Developers are interested in the details such as features and requirements, whilst product planners are also interested in market information and competitors. It is difficult to map this information automatically in practice. From a SW developer perspective it is on too high level.
Modification	None. Note it might be a capability that is difficult to realize.

Portfolio management

SPL-Capability	**Financial Product line scoping**
Usefulness	1/2
Rationales summary	This is very useful, especially for features. This also assures a sort of cleaning up in your pile of requirements and code, since requirements are left out or removed if these do not bring financial added value. These estimates are impossible to do that early in the process and therefore not reliable. From a SW developer perspective it is on a too high level (ignored).
Modification	This capability will need to be more ‘flexible’ with the financial calculations, i.e. any financial information that will add value to the decision-making should be included. More emphasis on the features too.

SPL-Capability	**Architectural product derivation**
Usefulness	0/3
Rationales summary	It is not useful if you want to have a good TTM; all products share the risk of the 'total/super' product and will need to wait till everything is done. It will make everything less efficient and slower, since extra dependencies might be created and

	other teams have to wait till the architecture is fully done. It seems that this would be a very static environment that only allows subsets of the 'total/super' product.
Modification	This capability will be left out.

Product line architecture

(The answers of the expert with more technical knowledge weight more)

SPL-Capability	**Reference architecture construction**
Usefulness	1/2 (one expert had no experience on this)
Rationales summary	It is useful to guide the design of the architecture or SW system and to identify the weak points. One expert indicates that not including quality requirements that early on makes them save time. Literature (Pohl, 2005) states that quality requirements should be considered at an early stage.
Modification	Emphasis should be put on early quality requirements consideration.

SPL-Capability	**Reference architecture construction based on scenarios**
Usefulness	1/2 (one expert had no experience on this)
Rationales summary	It is good practice to design the architecture using requirements, although other techniques than scenarios might be possible or better. One expert says they have no time for this.
Modification	Emphasis should be put on the input of requirements. The scenarios can be seen as guidelines to develop the reference architecture.

SPL-Capability	**Reference architecture construction based on variability**
Usefulness	1/2 (one expert had no experience on this)
Rationales summary	The risk if variability is not considered is that the usefulness of the architecture might be significantly lower than desired. It will limit the number of possible variants and products and increase modification costs. One expert says they have no time for this.
Modification	None

Organizational

SPL-Capability	**Domain & Application engineering**
Usefulness	0/2 (one expert had no experience on this)
Rationales summary	Does not make sense and the division into domain and application engineering does not seem feasible in practice. However, some sort of separation of developments concerns is advisable.
Modification	This capability will be left out. Despite being widely recognized in literature.

SPL-Capability	**Requirements engineering planning**
Usefulness	3/3
Rationales summary	It is important in order to know what to implement in which product and you need to plan this instead of letting everyone implement what he thinks is important. From a SW developer perspective, it is good to know that product management is planning what has to be done and not on gut feeling.
Modification	None.

SPL-Capability	**(Upper) Management Involvement**
Usefulness	1/2 (one expert had no experience on this)
Rationales summary	From a SW developer perspective, it is positive to feel the interest from upper

	management. However, they shouldn't concern themselves with keeping timetables only, instead they should also concern with the deliverables of the processes and their quality. Upper management does not know what is important and how to deal with it in a proper way. Hence, the task remains by project and product managers and they report to upper management. (The essence is to get upper management to be involved more, pro-active.)
Modification	Since the SPL-Capability of Domain & Application Engineering has been left out, this capability would need to be left out too for the reason it mainly focused around the Domain & Application Engineering capability. However, it will be rewritten with the experts input. Emphasis should be put on the deliverables and the quality thereof. Literature supports this, despite one expert stating that this should not happen.

Appendix D: Case company situational factors list.

REMOVED FOR CONFIDENTIALITY REASONS

Appendix E1: Product development interview instrument

Introductie

- **Product portfolio**

  Welke producten (software) worden door CCV Systems ontwikkelt voor:
  - CCV Holland
  - Internationaal (BE en CH)
  - (DE)

  Kan deze producten gecategoriseerd worden? Zo ja, in welke categorieën?

  Zijn deze producten totaal verschillend (nieuwe ontwikkeling) voor elk land of worden bestaande producten van een bepaalde land aangepast/gewijzigd voor een ander land?

  Zijn er momenteel producten die ontwikkelt zijn voor een bepaald land maar die ook gebruikt worden in een andere land?

Software Product Management

- **Algemeen Software ontwikkeling process**
  - Kunt u de algemene software ontwikkel process die CCV Systems hanteert beschrijven (bijv. opdracht type, opdrachtgever, functioneel + technisch rapport,
    enz.)?
  - Hoe ging het bij vorige projecten, bijvoorbeeld de laatste software release?
  - Worden er bepaalde software ontwikkelings methodes gebruikt, bijv. Waterval methode?
  - Kunt u de software architectuur (in het kort) beschrijven?
    Is er één architectuur voor alle software?
    Hoe is de software architectuur gedurende de afgelopen jaren geëvolueerd?

  - In hoeverre wordt er rekening gehouden met internationale producten (product dat 'inzetbaar' is in elk land) bij het ontwikkelen (architectuur, herbruikbare software, enz.)?

- **Details SPM processen (SPMCM)**
  - Requirements management
    - identificeren
    - verzamelen
    - organiseren
  - Release planning
    - prioritiseren
    - release definiëring en validatie
    - Scope change management
    - Build validatie
    - launch voorbereiding
  - Product planning
    - product roadmapping
    - (core assets roadmapping)
    - (roadmap intellegentie)
  - Portfolio management
    - markt analyse
    - product life cycle
    - partnering

- **Implicaties/betrekkingen van software ontwikkeling én software management processen op het meerstromen-land**
  - Main tree
  - TTM
  - S&I

Laatste vragen

- **Wat loopt volgens u goed bij software ontwikkeling én software management processen?**
    - Wat is makkelijk/wat kost minimale inzet?
    - Wat heeft verbetering nodig?

- **Wat loopt volgens u slecht bij software ontwikkeling én software management processen**
    - Wat is moeilijk/wat kost (te) veel inzet?
    - Wat mis u?