Master Thesis
Software Engineering
Thesis no: MSE-2010:24
September 2010

Adopting Software Product Lines

Guidelines based on the Maturity Levels of Organizations

Marcus Björklund and Jonas Hjelmar

School of Computing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 2*20 weeks of full time studies.

Contact Information:
Author: Marcus Björklund
E-mail: marcus.v.bjorklund@gmail.com

Author: Jonas Hjelmar
E-mail: j.hjelmar@gmail.com

University advisor:
Cigdem Gencel
School of Computing

School of Computing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

Abstract

Software Product Lines (SPL) is a relatively new way of working in software development. A SPL is an enforced architecture and a collection of components common for more than one product within a domain. Development using SPL utilizes tools and techniques for creating software systems from a common set of core assets in order to satisfy a certain market.
In this study we investigate how to proceed when transitioning to a SPL development approach by evaluating proposed and used adoption approaches, methods and frameworks. We performed a systematic literature review using three main sources; Compendex/Inspec, CiteSeerX and Google Scholar. The results are analyzed using a qualitative analysis technique called Recursive Abstraction where the results are iteratively summarized to extract the essence of the data.
A manageable collection of frameworks, methods and approaches are summarized as a starting point for a reader who wants to dig deeper into the subject. A set of guidelines is suggested for companies who are considering a transition to SPL development. We also investigate the link between SPL and organization maturity, with a focus on the benefits of combining a SPL initiative with a CMMI initiative. We conclude that the transition process should not be taken lightly; in most cases it should be made in incremental steps. There is a fairly standard approach to adopt SPL and there are a few frameworks that are commonly accepted. However, we also conclude that most research areas of SPL development lacks in validation. Concerning the link between SPL and CMMI we identify some PA's that are more important when considering SPL development and a few others that may be harder to execute.
We conclude that SPL benefits from process maturity and discipline as SPL development is process controlled and a lack in process discipline may cause corrosion of the SPL. A CMMI maturity level of Defined processes should be considered a prerequisite for a complete SPL practice. We could not find any indication that the organization maturity would benefit from SPL practices alone. Neither could we identify any drawbacks of having both an CMMI initiative and SPL transition initiative within the same organization.

Keywords: Software Product Line, Guidelines, Transition, Initiation, Adoption, Literature Review

Table of Contents

1 INTRODUCTION

Software development industry is more and more turning into an engineering discipline, which many researchers in the area strive for. One step in that direction that has been a big topic of research during the last decade is Software Product Lines (SPL). SPL is an architecture with belonging tools and techniques for creating software systems from a common set of shared core assets that will satisfy a certain market [NOR07, BOS01].

Documented benefits of successful SPL development are many and acknowledged. The major benefits are improved productivity and quality as well as decreased cost, labor and time to market [NOR07, BOS01]. With this in mind it is not hard to understand why many companies consider introducing SPL development practices. Taking the step from considering a transition to actually do it is not a small one. It involves large investments and often reconstruction of the organization. There are also numerous proposed approaches to this transition and how to work with different areas of SPL development. For a company who wishes to investigate the possibilities, the amount of articles and papers may seem overwhelming

1.1 Background

SPL consists in general of four parts; software assets, product decisions, production and software products [CHA04, BOS01, NOR07]. Based on the product decisions the software products are produced from the software assets.. The core assets may consist of requirements, source code components, test cases, architecture, specifications etc. Among these assets there are optional as well as changeable assets in order to make all the different products of the product line. The product decision is often illustrated with a decision model, which describes optional and variable features for the products in the product line. These decision are then used to determine the process of the production, how to compose and configure the assets to form products. The scope of the SPL is determined by the complete collection of software products that can be produced in the product line.

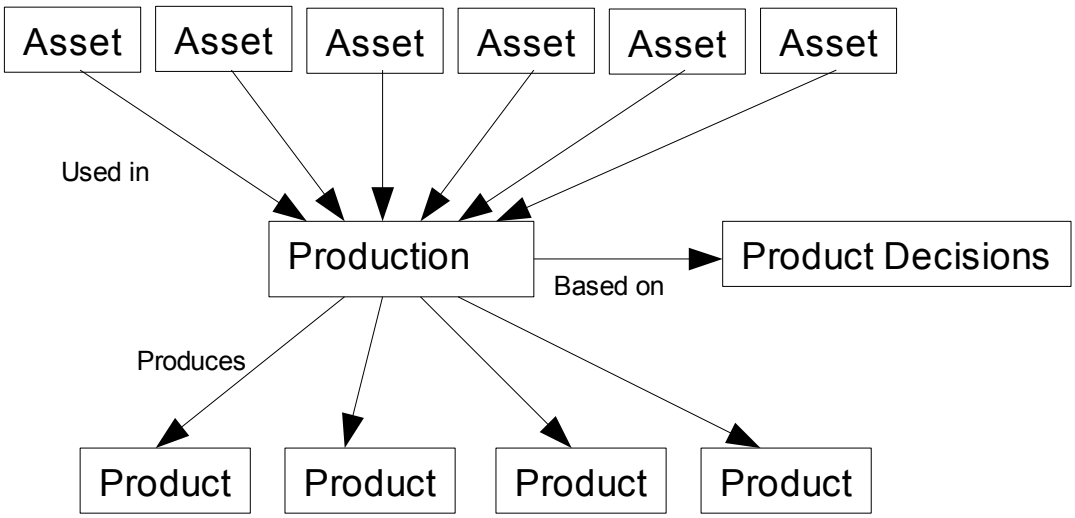


Figure 1: The four main parts of a SPL

In a SPL the variability is made explicit through variation points, which are the places in the architecture where the assets differ between the different products of the SPL [NOR07, BÜH05]. Variability must be handled throughout the lifetime of a product line, from planning and creation to finalization and maintenance. The challenge of SPL variability is to maintain one single implementation of each asset, even though it is used in different products that may put slightly different requirements on the asset. If successful this will result in a number of products that can be maintained as a single system [BÜH05, NOR07]. Insufficient variability management can have various consequences. One of the most serious is the total collapse of a SPL if too extensive modifications are made to the individual products so that they can no longer be considered instances of the same system [NOR07]. This is usually not something that happens overnight. It is rather the result of a long series of decisions to cut corners when a deadline approaches and make quick fixes in the individual products instead of entering cycles of analysis and redesign needed to make the corrections in the core assets. This phenomenon is usually referred to as corrosion.

There are several reasons why SPL is so hard to adopt. One big problem with SPL is the initiation because it can include a change of the organization's entire structure and process of development [NOR07]. Such changes pose a great risk and requires investments. It may be hard to convince management and employees and gain the necessary support needed for the initiative. Another obstacle is that investments have to be made up front. Although there are large potential long-term returns of the investments few companies, especially smaller ones, are able or willing to make the adoption in one leap.

The research field on SPL has been huge for the last ten years. There are a vast number of proposed methods for SPL engineering, many focusing on domain engineering. Several frameworks and numerous approaches on how to adopt SPL has been proposed. There are also a large number of case studies and experience reports available. These statements became apparent with a quick survey of the research field.

Although much research has been done and many frameworks, methods and techniques have been presented, there are very little research that investigate what has been done and what still needs to be done. To us the need for a systematic literature review became apparent. Specifically we identified a need to identify what had been done on the areas of how to adopt SPL development techniques and SPL adoption readiness. We also identified a gap in the research on what frameworks, methods and techniques etc can be considered the best or even suitable for companies that are new to SPL engineering. Managers of a company that considers a transition to SPL engineering would have to do extensive research in order to bring together the collective SPL knowledge needed to make the final decisions whether to proceed with the adoption and if so how to proceed.

The only systematic literature review we have been able to find related to this field of research was published by Khurum et. al. [KHU09] and is a systematic literature review on domain analysis methods. In their study they address the usability and usefulness of proposed methods. Our study include several other areas, including other method areas, and is angled towards SPL adoption and practitioners new to SPL concepts. We have not been able to find any systematic literature review that include SPL frameworks, adoption approaches or maturity.

Based on the results of the systematic literature review we wanted to construct a set of comprehensive guidelines for SPL adoption, as we could not find this anywhere. Related work to this goal were identified in the systematic literature review and used

to construct these guidelines. The related work include adoption readiness control tools and preparation methods like Product Line Potential Analysis [FRI04] and Product Line Technical Probe [SEI10b]. Other related works propose goals on what to achieve like the Framework for Software Product Line Practice [NOR07] and the Family Evolution Framework [LIN05]. In between these there are approaches for SPL adoption proposed by academia or reported by industry that intend to take a company from point A to point B. Our guidelines put all this together in a context and add reflections based other material gained through the systematic literature review.

1.2 Aims and objectives

The aim of this thesis is to provide information on SPL frameworks, methods and tools as well as constructing easy to follow guidelines for organizations that are considering a SPL approach. Secondary aim is to investigate the possibilities of combining a SPL with a CMMI initiative.

- Identify, evaluate and recommend frameworks for SPL development.
- Identify practice areas for SPL development.
- Identify, evaluate and recommend methods for practice areas.
- Identify, evaluate and recommend approaches to adopt SPL practices.
- Identify variables that affect the choice of SPL adoption.
- Investigate similarities between CMMI and SPL principles.
- Investigate if any CMMI process areas could be especially beneficial to SPL practices.
- Investigate how SPL practices could affect CMMI maturity level and process areas.
- Identify and evaluate research on SPL maturity.

1.3 Research questions

Our research questions in this thesis are put in three groups. RQ1 and RQ2 are aimed at collecting knowledge on SPL engineering. How to structure the way of work and which methods there is to use. RQ3 and RQ4 concerns the transition to SPL and how to analyze if an organization is capable of making the transition. This knowledge will be the basis for our guidelines and a starting point for companies to continue from. RQ5 are investigating if there exist a connection between Software Product Lines and a CMMI process improvement initiative and if there could be any advantages of combining them. The final research question are answering if the SPL practices can be categorized according to maturity to be able to use this when constructing our guidelines. The answers to the four first research questions will together with lessons learned in case studies and experience reports be the basis for the guidelines. The methodology are visualized in Figure 2.

- RQ1. How should the SPL practices be structured?
    - RQ1.1. What available frameworks for SPL development are there?
    - RQ1.2. What are the differences between the found frameworks?
    - RQ1.3. Which of these can be recommended?
- RQ2. What methods should an organization consider for their SPL development?
    - RQ2.1. Which are the specific areas of software product lines?

    - RQ2.2. What available methods are there for each area?
    - RQ2.3. Which of these can be recommended?
- RQ3. How does an organization adopt a SPL approach?
    - RQ3.1. What available adoption approaches are there?
    - RQ3.2. Are there any differences between the adoption approaches proposed by researchers and those used in industry?
    - RQ3.3. What are the differences between specific adoption approaches?
    - RQ3.4. What variables affect the choice of adoption approach?
        - RQ3.5. What can be learned from industry case studies?
    - RQ3.6. Which of the found adoption approaches can be recommended?
- RQ4. What are the available methods on analyzing an organization's readiness towards adopting a SPL approach?
- RQ5. What are the consequences of combining SPL development and CMMI process improvement?
        - RQ5.1: Are there any similar procedures or concepts between the two initiatives?
    - RQ5.2. Does the CMMI maturity levels affect the SPL adoption process?
    - RQ5.3. Are there any CMMI process areas that have a greater benefit for SPL development than for single system development?
    - RQ5.4. Are there any CMMI process areas that are more difficult to apply in SPL or has a negative effect on SPL development?
    - RQ5.5. Can the introduction or improvement of SPL practices improve the organization's process maturity as specified by CMMI?
- RQ6. Can SPL practices be categorized into maturity levels?

1.4 Research Methodology

The nature of the work presented in this thesis is exploratory. As the SPL research field has expanded quickly during the last decade we wanted to make use of already existing research. As we started out we did not know what we would find or what we could use. Because of this we needed to use a rather wide search pattern.

A systematic literature review is a suitable approach when exploring what research has already been published. It can be made very specific to target a small niche of articles or like in our case be made more generic to capture many aspects of a certain area depending on the search strings used.

The strength of systematic literature reviews is that it is a structured and controlled way of collecting data. With a good design and using good sources at least the majority of relevant articles can be identified. We started by doing a thorough prestudy where sources and keywords were examined. Based on the results we planned and conducted our systematic literature review (SLR) according to the guidelines given by Barbara Kitchenham [KIT04]. In Chapter 2 the detailed description of the method used and a presentation of the SLR results are provided. The Literature Review Protocol can be viewed in [Appendix A]. In addition to the database searches a manual scanning of the reference list of collected articles was conducted.

We surveyed the previously published research and case studies and providing a manageable set of frameworks, methods and techniques suitable for SPL adoption and

then defining guidelines for SPL adoption based on already existing empirical evidence would benefit software organizations more.

We also included the topic of SPL maturity which we combined with available material on organization maturity in an attempt to aim our guidelines at different classifications of organizations. We chose to use CMMI for this purpose in our work. We made this decision based on three facts. Firstly, CMMI is accepted and used for classifying organization maturity in software developing companies all around the world. Secondly, CMMI is developed by the Software Engineering Institute (SEI) which is also the institution behind the most extensive framework for SPL development. And thirdly, ISO-TickIT that were also considered as it has similar characteristics to CMMI and considers process definition, discipline and improvement, was undergoing a major reconstruction as this project started. So instead of working with the old version or the partially publicized new version of ISO-TickIT we decided to put it to future works and use CMMI instead.

When the articles had been collected they were categorized, as a first step towards analyzing the data. As we had ended up with a large number of articles touching on various topics concerning SPL and organization maturity this categorization was considered necessary and made the data analysis less complicated. The process of categorization is explained in detail in section 2.1.5.

There are other techniques that could have been used instead of systematic literature review like the snowball technique where the search is started with one article and continued from the reference list. This is then iterated with each article of interest. To use the snowball technique as the main methodology the area of investigation should be rather constrained to ensure that all relevant material is included in the references of the others. We had several areas of interest that we wanted to explore and we could not know beforehand how well the articles referenced each other. The technique also need to have a good starting point which we did not initially have. We chose to exclude the snowball technique as our methodology but in reality we had a limited snowball effect when we performed the manual scanning of the reference list mentioned above.

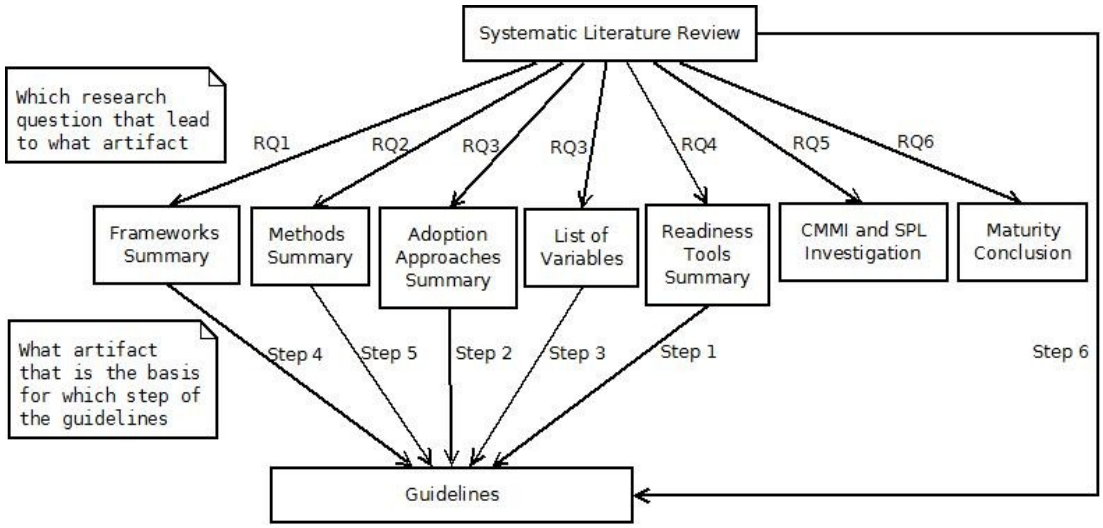


Figure 2: Abstract View of Methodology and Artifacts

For the data analysis we chose to use a qualitative analysis technique called Recursive Abstraction (see Figure 3).

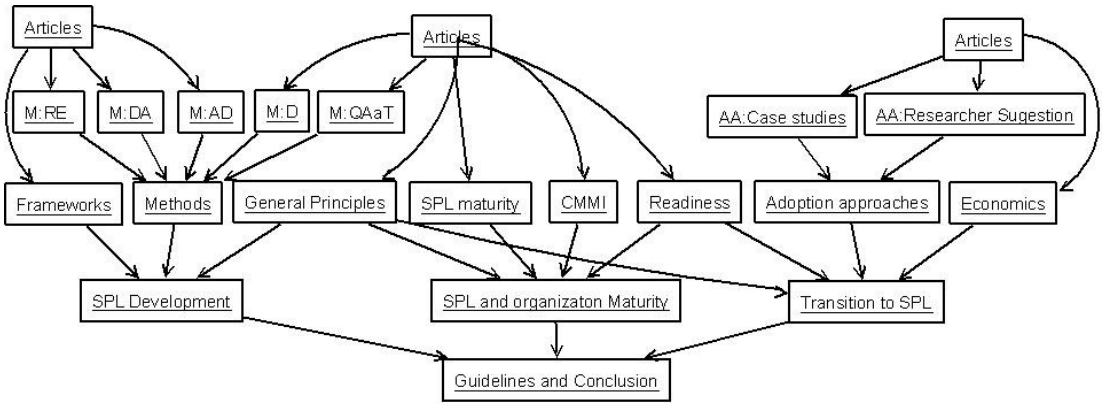


Figure 3: Recursive Abstractions

Recursive Analysis is a technique that is good at handling large quantities of textual data of qualitative nature. In contrast to Grounded Theory, which is another technique, Recursive Abstraction does not rely on coding of the data.

In recursive abstraction, the original data is first summarized, and then the summary is summarized further and so on. The end result is a compact summary that includes the essence of the studied field. We found Recursive Abstraction appealing as it allowed us to summarize each research area that we were interested in independently before using the results for further analysis as we combined these areas.

We started by summarizing each article. The article summaries were then used to summarize each category mentioned in the previous paragraph. In those cases where sub categories existed, these were summarized first and then summarizing the summaries of the sub categories derived the summaries of the main category. As we had three main areas of interest that we investigated they became the next abstraction level quite naturally. These areas were the SPL development, the transition to SPL development and the maturity of SPL and organization. See Figure 3 for a visual overview.

Within this thesis there are discussion and conclusions made at several of these abstraction levels. For example; the summaries of the frameworks and methods categories were analyzed directly and the discussion of these areas can be found in Section 2.2.1.1 and 2.2.1.2 respectively. Section 2.2.2.2 where the combination of SPL and CMMI is explored is on the other hand on a higher abstraction as several categories had to be combined and analyzed including the frameworks, CMMI, SPL maturity and general principles categories.

Through analyzing the collected data in the Frameworks, Methods and General Principles categories we answer RQ1 with subquestions in our discussions of this research area in section 2.2.1.

In Section 2.2.2 we dedicate the first subsection to answering RQ2 as we analyze the research on SPL maturity collected through our literature review. The following subsection of Section 2.2.2 discusses the relationship between SPL and CMMI initiatives. Although largely inconclusive it answers RQ4.

RQ3 including subquestions is answered in Section 2.2.3. The chapter is opened with a description of the terminology used in SPL adoption research derived by

analyzing proposed and used adoption approaches. This is followed by a section where we describe three classifications of adoption approaches that we created after analyzing the collected research and deriving the terminology. These classifications are then used in our guidelines in Section 2.2.4. Section 2.2.4 is then ended with a discussion of proposed and used approaches for SPL adoption as well as the readiness to perform such an initiative. The discussion on adoption readiness provides the foundation for answering RQ5.1.1 and RQ6.1.1 and is mainly based on our analysis of readiness assessment tools gathered in our literature review.

RQ5 and RQ6 are answered in Section 2.2.4 as we provide our guidelines on how to initiate the transition to SPL engineering. These guidelines are based on the collection and analysis made in the previous chapters as well as analysis of case studies and experience reports collected through our literature review. In Section 2.3 we present short answers to our research questions and belonging subquestions and links to full discussions of each question.

In addition to our guidelines we provide a small collection of summaries on frameworks, methods and adoption approaches that was deemed to be potentially useful for a company that are considering a SPL adoption. The intention of this collection is not to necessarily to provide the best works as the term “best” is very subjective and the “best” methods may be very complex and perhaps not suitable to small companies or companies that are new to SPL. The collection is intended as a starting point for continued investigation by anyone interested in learning more of a certain area and only include a very brief description and evaluation of the works as well as references to more complete material. These summaries can be found in the appendixes B, C, D and F.

2 Software Product Line Adoption

2.1 Systematic Literature Review

2.1.1 Data sources and search strategy

For the systematic literature review we had a three phase search strategy. In the first phase we conducted a database search. In the second phase we manually scanned the reference lists of the included articles and in the third phase we identified and accessed web resources.

2.1.1.1 Phase one

We used three different databases search engines as data sources in our systematic literature review. The databases used were Compendex combined with Inspec through Engineering Village (www.engineeringvillage2.org), Google Scholar and CiteSeerX. The combination of Inspec and Compendex results in a thorough search of science and engineering journals and conference proceedings. CiteSeerX was chosen as a complementary database to Compendex/Inspec since it's main focus is Computer Science. The choice of Google Scholar is covered below.

Several other databases was considered but was excluded since either they used the same sources as our chosen databases or was focused on a different field of science, eg. medicine, biology etc. During our prestudy we ran searches using the same search strings on engineering village, IEEE explore, ACM digital library and Springer link. We found that the search on engineering village had a complete coverage of the results on the other three databases. Although this was a sample of the complete search we were to make, we made the assumption that searching IEEE explore, ACM digital library and Springer link individually was not necessary. As a precaution and as not to include a validity threat we used the mentioned databases when collecting primary studies from those sources. While doing this we manually scanned the titles of the other articles in the conference proceedings or journals that we collected articles from. If we found that our first search had missed articles that were of interest we would have had the option of searching these databases individually as well. As we could not find any articles that we had not already found we made the assumption that our first assumption was correct and that these databases did not need further searching. Other than the precaution procedure mentioned, manual search of journals and conference proceedings was deemed unnecessary since test runs of the chosen databases included all journals and conference proceedings we could identify as of interest.

The use of the Google search engine as primary source of gray literature was considered but found unsuitable. As the research areas covered in our systematic literature review is both large and diverse a Google search gave many millions of hits. Even a rather large sampling strategy would have had a very poor coverage. Another drawback we identified with Google is that Google ranks the hits based on the number of links to that hit. This can indicate that recently published resources will not have a very high rank and end up far down the result pages. As older gray resources are more likely to be referenced by other articles it is the newer additions that we were most interested in finding this way. Due to this we choose to use Google Scholar as a primary source instead. Google Scholar has a wider search pattern than other scientific

databases and includes alternative sources of gray literature. At the same time the search is limited enough for us to review the entire search result, thous avoiding a sampling strategy. A problem with Google Scholar that discourage its use is that the advanced search function is rather limited. This problem was overcome by breaking down the search strings to the maximum allowed complexity before the searches and then merging the results according to how the search strings were originally constructed.

Our choice not to use Google as a main data source for gray literature may have caused us to miss some sources of gray literature. But as we have added gray literature by the means of Google Scholar, reference scanning (see 2.1.1.2 Phase Two), identification of prominent research contributors and specified Google searches (see 2.1.1.3 Phase Three) we believe we still have a rather good coverage of gray literature. And as we would have had to use a sampling strategy with a rather limited coverage on Google as a main data source due to the sheer number of search results, it is pure speculation if Google would have had a better coverage of gray sources.

A decision was made to limit the search to the last decade, e.g. 2000-2010, as the SPL research field is fairly new. Granted that there are earlier research, mainly on domain engineering performed in the late 90's, but many of those works are according to our prestudy quite well known and referenced in later research. We had at this point already identified Khurum et. al. [KHU09] systematic literature review of domain analysis methods published prior to 2007, and we were confident that this area would be sufficiently covered. Thus earlier research of relevance was included by manual scanning of the reference lists of selected articles (see 2.1.1.2 Phase Two).

The search terms used was derived during the prestudy and test runs of the databases. The search terms was then separated into four groups (Table 1) and the search strings was constructed using boolean AND statements between groups and OR statements within groups. Three searches was conducted on each database where the search string was constructed by combining the search terms form group 1 with one of the other groups as can be viewed in Table 2. The results were then merged and the duplicates removed.

2.1.1.2 Phase two

Phase two of our search strategy was conducted after the first round of primary study selection had been conducted. The reference lists of the selected studies were manually scanned for articles of interest that might have slipped through the database searches. The references included quite a few web resources and articles of gray literature. In this phase we used the snowball technique. An included reference or web resource could reference yet another source of interest that was in turn included.

2.1.1.3 Phase three

As the final part of our search strategy we used the already included studies as a guide to identify authors, institutions, concepts etc that was of interest to this field of research. Guided by this we tried to identify and access web resources and gray literature using Google and more specific search strings compared to the previous database searches.

Table 1: Groups of search terms

Group 1	Software product line/lines, software product family/families, software product factory/factories.
Group 2	framework(s), model(s), practice(s)
Group 3	maturity, readiness, preparedness.
Group 4	initiation, adoption, introducing.

Table 2: Search string construction

Search 1	At least one term of group one and at least one term of group two
Search 2	At least one term of group one and at least one term of group three
Search 3	At least one term of group one and at least one term of group four.

2.1.2 Study Selection

2.1.2.1 First refinement

For the first refinement the titles and abstracts of the initial search result was compared to a checklist of inclusion/exclusion criteria (Table 3). The checklist was gone through in a top-down manner. When one criteria was considered true the article was included or excluded accordingly and the rest of the checklist was ignored. If the bottom of the checklist was reached without the article matching any criteria the article was given the benefit of the doubt and was included for the next refinement.

Some of the checklist items could not be answered from title and abstract alone. These checklist items was not evaluated during the first refinement and were considered first at the second refinement. Similarly, gray literature may be missing title and abstract. In that case the source was included for the second round of refinement by default.

This process was conducted by both authors respectively and if disagreement occurred on inclusion or exclusion the article was included for further analysis. A full list of the included articles with complete references from this refinement can be viewed in Chapter 7: References.

Table 3: Inclusion/Exclusion criteria

Inclusion/Exclusion Criteria	If true
Not written in English.	Exclude
Redundant (there exists later versions of the article etc.)	Exclude
The subject is not concerning SE and SPL.	Exclude
Concludes an artifact as a SPL success factor.	Include
Includes lessons learned or suggestions based on case study.	Include
Includes comparison with or references to CMMI	Include
Discusses a model, method or framework for SPL.	Include
The subject is concerning SPL maturity, readiness or categorizing of SPL organizations	Include
Discuss a method or approach for initiating SPL.	Include
Not well referenced (empty statements)	Exclude

2.1.2.2 Second refinement

Articles that could not be excluded after the first refinement was obtained in full text and underwent a second refinement. The content of the articles was viewed in full and compared a second time to the checklist of inclusion/exclusion criteria (Table 3). The procedure was the same as for the first refinement with the exception that by now it was possible to evaluate gray literature and all checklist items.

2.1.3 Quality Assessment

The quality of the studies was categorized as satisfactory or unsatisfactory. A satisfactory article should contain at least one of the attributes in Table 4. Each article was reviewed by both authors of this thesis respectively and a note of the quality was attached to each article. If disagreement occurred the article was debated until both authors agreed.

Table 4: Criteria for defining satisfactory studies

A clear definition of at least one model or framework for SPL.
A scientific and well performed case study on a organization using a certain model or framework for SPL.
A clear definition on SPL maturity/readiness.
A clear definition on maturity of artifacts of a SPL.
A description of how to categorize SPL in a novel way.
A clear definition on at least one approach for initiating SPL.
A scientific and well performed case study on a organization which adopted the SPL approach.

2.1.4 Study Selection Results

The first search string on the three databases resulted in 2182 hits. The second search string added 81 hits and the third search string added 669 hits. This made a total of 2932 hits on the database searches. After the first round of refinement using the inclusion/exclusion checklist on title and abstract there was 272 articles to be retrieved and read in full. Out of these 272 articles, 187 came from the first search string, 34 from the second and 51 from the third. After the second round of refinement the total number was decreased to 146 articles, out of which 102 was derived from the first search string, 14 from the second and 30 from the third. Another 58 articles and Internet resources was then added after scanning the reference lists of the selected articles increasing the total number to 204. A list of the completed set of articles can be viewed in chapter 7. References. A graphical view of the search results are shown in Figure 4.

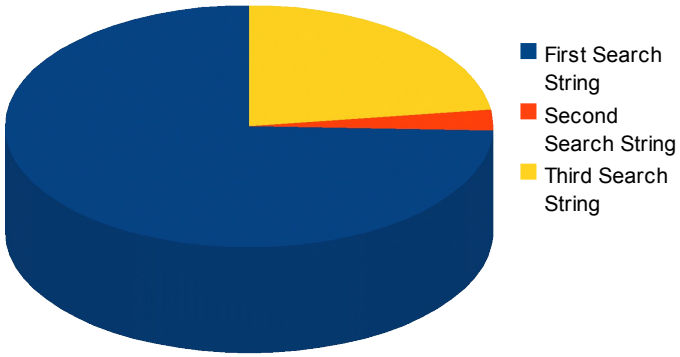


Figure 4: Distribution of search string results

2.1.5 Data extraction

Articles that made it through the refinements were categorized into type, and when necessary subtype, of article topic. Some of the main types such as Frameworks, Methods and Adoption Approaches was identified as early as in the prestudy as researchers often uses these terms to describe their own work. Other types such as Readiness, Maturity and CMMI was identified when the literature review was constructed as these are specific areas that we were interested in. Finally there was a few articles that the initial analysis showed could be of interest to this thesis but did not fit into any of the previously mentioned types. Some of them concerned economical aspects and was thous classified as such. The rest concerned SPL engineering principles but did not specify a specific framework or approach. These articles was categorized as General Principles. As some of the types such as Methods included a large number of articles it was decided to divide these further into subtypes. These subtypes such as Domain Analysis and Architecting and Design were identified during initial analysis of the articles as the focus of the articles were identified.

When one article could fall in more than one category as was not uncommon among methods, it was put in the category where it had its main focus. A method such as Feature-Oriented Reuse Method (FORM) [KAN98] focuses on how to perform a feature domain analysis and then include how to transform the result into an architecture. It could fall within both the Domain Analysis and the Architecture and Design categories but as the focus of the method is on the feature oriented domain analysis it was categorized as a Domain Analysis method. Because of this overlap on some of the categories it was impossible to view them as completely independent and this had to be taken into consideration when the analysis was performed, but this categorization did help us perform the analysis according to the principle of divide and conquer. The categories used can be viewed in Table 7.

The articles were summarized in plain text in accordance with our chosen methodology and discussed between the authors of this thesis. For the article summaries we used an extraction form with the headlines shown in Table 5. For the evaluation we mainly looked at two attributes, level of validity and the level of method/process description provided. The validity levels was defined as presented in Table 6. The level of description was a estimated measure of how well the method or process could be repeated by a reader based the descriptions provided in the article. A method or process with step by step descriptions would be considered to be very well described while a case study where results where provided in detail but not the road to the result would be considered of low descriptive level.

When summarizing article categories or areas of interest, as described in Section 1.4: Research Methodology, we started out with a blank form that was filled in as the analysis progressed under conditions similar to a small workshop.

Table 5: Article summary data extraction headlines

Source	Author	Purpose	Description	Evaluation

Table 6: Levels of validation

Level of validation	**Definition**
None	None
Low	Based on existing models and frameworks and/or validated by a hypothetical example or internal case study.
Medium	Gained validity by some usage in industry.
High	Used in industry with quantitative and qualitative data provided

Table 7. List of categories used and the number of articles in each category.

Category	**Amount**
Frameworks	18
Methods Requirement Engineering	4
Methods Domain Analysis	49
Methods Architecting and Design	30
Methods Development	5

Methods Quality Assurance and Testing	5
Adoption Approach Case Study	23
Adoption Approach Researchers	20
Readiness	9
SPL Maturity	11
CMMI	9
Economics	6
General Principles	15
Total	204

The next chapter summarizes the findings and discussions surrounding the extracted data. Some of the summaries were rewritten and used in this thesis as introductions to frameworks, methods and approaches that companies according to us may find useful.

2.2 Data Analysis and Results

In this section the analysis and the findings of the systematic literature review are analyzed and discussed. The focus is on what has been proposed by researchers or done in industry to this date. This is done in order to help any company considering a SPL initiative to find relevant information, such as methods, approaches and frameworks available and which ones that have worked before. As the data collected through our systematic literature review was mainly qualitative we analyzed the data by qualitative means [MIL94, DEN05]. The data was summarized and discussed recursively, a technique called Recursive Abstraction. Please refer to the Research Methodology section in Chapter 1 for further details.

The section is structured as follows: first there is a discussion on SPL development (Section 2.2.1). This discussion includes evaluation of methods and frameworks in order to answer RQ1 and RQ2. After that follows a section (Section 2.2.2) on SPL and organization maturity answering RQ5 and RQ6. Section 2.2.3 covers the topic of SPL adoption approaches and strategies, i.e. the transition to a SPL development approach. This answers RQ3 and RQ4. The combined results of the four first research questions are the guidelines presented in Section 2.2.4

2.2.1 The Development

The project lifecycle of SPL development is not that different from single system development unless you choose to make it so. The development of assets can be viewed as a series of mini projects were each asset, or a small collection of assets, goes through the development phases of requirement engineering, design, implementation and testing etc. Similarly the derivation of a product goes through the same lifecycle as single system development, although the coding part of the implementation phase is largely substituted by the choosing and linking of already developed assets. Hence, the main body of SPL development practices does not differ that much from single system

development. We will return to this statement at several occasions when we discuss specific development areas.

2.2.1.1 Frameworks

At the organizational and managerial levels there are quite a few differences between traditional single system and SPL development. In order to help practitioners incorporate practices that are considered essential there are frameworks that the organization can use. Following one of these frameworks should help in establishing a SPL environment [NOR07]. Whether it is an internally developed framework within a company or a publicly available international framework, the most important thing is that the framework is customized for the specific organization. A good framework should state which activities to include, describe them, their purpose, possible issues, what to consider and what happens if this activity lacks in execution. Before starting to customize a framework for the company needs, some consideration should be put into what framework to start out with. Pick the one that fits the organization the best and then customize it. Some of the most applied and validated frameworks are discussed below [Please see Appendix B: Framework Summary for more details].

2.2.1.1.1 Frameworks Summary

The most famous of the frameworks for SPL is probably the Framework for Software Product Line Practice [NOR07], which was developed by the Software Engineering Institute (SEI), the creators behind CMM, henceforth the SEI Framework. This framework has been updated several times since it was created in the late 90's and are now available online as version 5.0. The SEI Framework is one of the most extensive frameworks and covers all areas needed for a successful SPL practice. Such as domain, scope and market analysis, asset creation and extraction, architecture creation and evaluation, technical and organizational planning and configuration management [NOR07]. One thing missing according to us in this framework is a suggestion on the order to perform the activities in. The practice areas of the framework often overlap and many are most likely to be carried out simultaneously.

The Generic Product Line Process Framework [VEH00] and the Product Line Engineering Practices (PLEP) Model [COA05] also covers all areas but not as thoroughly as the SEI Framework. The Generic Product Line Process Framework is a product of a comparison of four other frameworks including an early version of the SEI Framework. This framework is a bit brief but it includes all strong parts of the compared frameworks. However, in our opinion it requires a bit more work.

The PLEP model is another framework that is based on the SEI Framework as well as the VRAPS model [DIK01]. VRAPS stands for Vision, Rhythm, Anticipation, Partnering and Simplification which means that VRAPS is about creating the architecture with a clear Vision, a good Rhythm, well Anticipation, uncomplicated Partnering with stakeholders and Simplification of environment. As SPL development is usually very architecture centric this integration should prove effective. The PLEP Model is created to incorporate well with existing development methodologies and system engineering standards and has grouped the practice areas based on activities to remain consistent with the CMMI.

Yet Another Make (YAM) [JAI06] was created a decade ago and it has evolved based on adoptions made in the companies that have been using YAM. It groups the activities based on needs which makes it easy to see what activities an organization

needs based on what the organization wants to accomplish. Although it has been considered for SPL development, YAM is aimed at software development at large. Thus it lacks somewhat in SPL specific practice areas.

Most of the frameworks are created by researchers in the field of software engineering. An exception is the BigLever's Software Product Line Lifecycle Framework [BIG09] which was created inside a company and is unfortunately not publicly available. A benefit of industry created frameworks that are still in use is that they obviously work, or they would not continue to be used. Drawbacks include that this type of framework might be too specialized for the original organization so that it will be impossible for other companies to adapt it for their situation. Another drawback is that since it is not easy accessible most companies will not even have the opportunity to view it.

These frameworks all concentrate on what to do and not so much on how. The Integrated Process (TIP) [KIM06]a on the other hand have integrated regular SPL practices together with Component Based Development (CBD) and Model Driven Architecture (MDA). This make this framework quite useful and hands on since the steps on how to design assets and components, how to incorporate models in the development and how to implement and customize the products from assets are thoroughly explained.

The last framework we want to mention here differs from the rest, it is basically a evaluation framework. First versions of the framework is simply called BAPO, the latest version also uses the BAPO dimensions but is called the Family Maturity Framework (FMF) [LIN02, LIN0, LIN05]. FMF gives a good overview of the entire SPL effort by evaluating the maturity level of your Business, Architecture, Process and Organization. If one part has lower maturity then the others, then investigating why, can lead to findings of issues or things that has been overlooked. This framework is thus useful after you have established an SPL and want to evaluate it. It could be good to use the definitions of the maturity levels as goals for that dimension of the organization.

2.2.1.1.2 Discussion

The discussion on similarities and differences are partly based on Table 8. Most of the frameworks discussed have some sort of practice areas or activity groups that describes what you should do. These practice areas are often grouped in larger categories to get a better overview. Another already mentioned similarity is that almost all of the frameworks only discusses what to do and not how. This could be both a benefit and a drawback. A benefit cause the framework focuses on incorporating the correct activities, but a drawback cause the company might choose a method or tools to do a certain activity that is inadequate.

One discussion that occurred during this framework evaluation was if the grouping of the activities mattered since there is almost the same activities just arranged differently. But the conclusion was that the order in which these activities are done will change the outcome. With further discussion we found that this is one of the few differences between the frameworks since they have almost agreed on what activities that should be included for successful SPL practice. Many of the excluded frameworks only stated what activities to include and did not explain them. A good framework should contain more information on the activities, why to perform them and also give suggestions on methods and tools to use.

It is our belief that a framework or any proposed solution should have been updated and evolved at least a couple of times since the first original idea. The reason for this is that it is in our opinion extremely difficult to get a theory completely correct in the first version. A theory that effects a whole organization in the way a framework does should have been applied and possible reconstructed for several iterations before it can be considered complete.

Table 8: Framework Comparison

Framework	Source	Validation	Provided Assistance	Version	Comment	Practice Areas
The SEI Framework	[SEI07]	Medium High	Illustrations and tool support	Optimized	No additional comment	Complete
The BAPO Framework	[LIN04]	Medium	None specific	Evolved	Used for evaluation of a SPL	Not applicable
The RSEB Model	[JAC97]	Medium	None specific	Evolved	Outdated	Lacks management
The Generic Framework	[VEH00]	Low	None specific	Initial	Based on FSPLP, RSEB SPICE and DsE	Complete
The PLEP Model	[COA05]	Low	VRAPS Guidelines	Initial	Based on SEI and VRAPS	Complete
The Integrated Process	[KIM06]	None	Method and tool support	Initial	No additional comment	Complete
Yet Another Make	[JAI06]	Medium	None specific	Nearly Optimized	Not specifically for SPL	Lacks in SPL processes
The BigLever Framework	[BIG09]	Medium High	Tool support	Nearly Optimized	Not publicly available	Unknown

Validation: *None, Low – Based on existing models or frameworks only, Medium Low – Internal case study and example provided and/or mentioned and used by others, Medium – Well known, used and/or used as a basis for other frameworks, Medium High – Used in industry, High – Used in industry with quantitative and qualitative data provided.*

Provided Assistance: *Can be in form of examples, methods, tools or guidelines.*

Version: *Initial – First draft of framework, Evolved - Evolved once or twice after first draft, Nearly Optimized – Evolved several times and updated based on cases, Optimized – Evolved multiple times and optimized based on cases and lots of experience, Finalized – Final version.*

Comment: *Additional comments on the framework, often some flaws.*

2.2.1.2 Methods

When it comes to what methods to use during certain phases of development, there are a lot of different methods developed for some areas and practically none in others. To increase readability of this sub section we have structured it according to the waterfall model with a slight adaption for SPL. Starting with requirement engineering, domain analysis and architecture and design then moving onto implementation, product derivation and testing. At the end of this sub section there is a short summary.

2.2.1.2.1 Requirements engineering

There is little research specifically aimed at requirements engineering for SPL. Perhaps this is because it is largely included in domain analysis, or that it is not that different from requirement engineering in general.

The techniques used to elicit requirements for a SPL are the same as those used for single systems development, e.g. interviews, inspections of previous work, observations etc. After the initial elicitation and negotiation of requirements, which is usually done for each specific product, the process usually steps over to the area of domain analysis. As a result the requirements are often talked of as an input to domain analysis methods, but there is almost no mention of the requirement engineering itself.

Another way of using requirements in product lines is to use them as a blueprint for automatic product derivation. This requires a mature and rather complete SPL as well as a predefined and more formal way of writing the product requirement specification. Unfortunately we have found no validation of any such method.

The closest thing to a specific requirement engineering practice for SPL that we have found was suggestions to write a generic requirement specification from which each system can be derived as a subset of the specification [FAU01, MOO05].

We conclude that the lack of interest in specific requirement engineering methods for SPL is due to a combination of two things. Firstly, the main body of requirement engineering techniques form single system development does still apply to SPL development. Secondly, the differences that do exist are considered part of domain engineering.

2.2.1.2.2 Domain analysis

Domain analysis is a core practice of SPL development and is also one of the things that more anything else difference SPL development from single system development. It has also been identified as one of the critical factors to SPL success [CLE01] and were the focus of most research on product lines. Hence the amount of available material on different aspects of domain engineering and proposed methods and solutions is huge. Although most proposed solutions claim to be effective few of them have been validated with real evidence to work in industry [KHU09].

A definition of domain analysis is "the process by which information used in developing software systems within the domain is identified, captured, and organized with the purpose of making it reusable (to create assets) when building new products" [AME01]. The purpose of domain analysis is hence for the practitioner to firstly understand a system or set of systems on an abstract level and secondly to aid in the decision making of what to include in the product line.

The specific practices of domain analysis can be divided into modeling and scoping. Modeling activities are identified by [AME01] as : Conceptual modeling, Requirements modeling, Commonality and variability modeling, Domain modeling, Feature modeling, Scenario modeling and Design mechanism capture.

Similarly, the scoping activities are identified by [AME01] as: Domain scoping, Product line scoping (or Product portfolio scoping) and Asset scoping.

Not all activities are always used in and often the activities overlap. For example, commonality and variability modeling focuses on the identification of similarities and differences between requirements of the different systems in a SPL. These similarities and differences are often identified as features, which are modeled in the feature modeling activities and possible design patterns associated with the feature is captured in the design mechanism capture activities. Requirements modeling and scenario modeling can substitute feature modeling or they may be used together as compliments. From this example it is clear why many methods for domain analysis does not clearly separate these activities. For a more extensive definition on the specific activities mentioned above, please refer to [AME01].

In the category of modeling methods there are a large variety of methods. In general, domain modeling methods can be divided into feature modeling, use case modeling and requirement modeling methods. Some of the more complete methods use a combination of these techniques but the focus is usually on one of the three.

The majority of modeling techniques belong to the feature modeling category. Feature modeling techniques group functionality together into features and model these features relations to each other, most often in a hierarchical tree structure. It is important to note that this structure has nothing to do with the architecture or design of the system. Often the differences between feature modeling methods is a matter of notation and tool support, this makes it hard to recommend one over the others as available and familiar tools as well as taste in notation may be very individual to each company. In [Appendix C: Method Summary] there is a brief description of some of the more referenced methods. For a good overview of features, feature modeling concepts and some useful practical guidelines that can be applied independently of chosen feature modeling method please refer to [LEE02].

Among the methods to be classified as scoping methods there is one that clearly stands out, and that one is PuLSE-Eco. The PuLSE methodology is briefly described in [Appendix D: The PuLSE methodology] and is one of the few methods that can truly be recommended since it is one of the few well validated methods.

As mentioned above, there is a huge amount of research being done in the area of domain engineering and analysis and the level of validation is quite low [KHU09]. With this in mind it is very hard to recommend any methods or solutions for domain engineering. However, in a recent survey of solutions presented between 1998 and 2007 [KHU09], the authors identify 36 solutions that they categorize to be both usable and useful according to the authors criteria, with a varying degree of validation. [KHU09] concluded in their systematic review that evidence to support claims of validity as well as usability and usefulness are lacking. We agree to this conclusion, according to our review some papers completely lack validation, others claim validation but presents no evidence of it and not a single solution presents quantitative evidence as validation.

2.2.1.2.3 Design and Architecture

Effective system design is an important aspect of all software development, and more so for SPL development. The SPL architecture dictates how the products are placed together and how well the SPL can adapt to changes in requirements over time. The assets used in the SPL has to be designed for reuse in future products with requirements that is not known at the time of the initial design. And to make it worse, the design will be hard to change ones the SPL is put to use [THI02].

Most research on design aimed specifically at SPL deals with the architecture as this is considered the central part of SPL practices. Most agree on the importance of the SPL's common architecture and propose various ways of engineering it by focusing on different aspects.

Matinlassi et. al. [MAT04a] did a comparison of five architecture design methods for SPL in 2004. The methods compared were Component Oriented Platform Architecting (COPA), Family-Oriented Abstraction, Specification and Translation (FAST), Feature-Oriented Reuse Method (FORM), Komponentenbasierte Anwendungsentwicklung (KobrA) and Quality-driven Architecture Design and Analysis (QADA). Matinlassi et al. concludes that these five methods has different focus and does not compete with each other. Thus either one could be used depending

on what suits the current project [MAT04a]. As we did our own evaluation of these methods we found that COPA, FAST and KobrA are rather complete SPL development methods and includes so much more then the architecture creation. FORM is a domain analysis method that extends into architecture and design and QADA is an approach for adopting SPL development. All of them include architecture creation and some are architecture centric.

In addition to the above mentioned methods there are quite a few proposed methods centered on architecture creation. Together with domain analysis this is the most explored area of SPL engineering. Unfortunately the design and architecture area share the domain analysis area's lack of validation.

2.2.1.2.4 Asset development

SPL engineering is about analyzing and planning for reuse. It is not surprising that the initial phases of the development lifecycle has been given focus and priority. There are also some research on the verification and validation of models and assets, as well as on how SPL assets are used to derive finished products. But in between these areas lies the actual development and implementation of those assets, the realization of models and design to be used in the product derivation. In spite of our efforts we have found almost no research or experience reports concerning this area of SPL engineering.

[GAC01] provides a quick introduction to a variety of fairly common implementation techniques that can be used to enhance variability at code level. The techniques are given a short description and pros and cons are discussed and compared with each other. The support of the techniques are also investigated in a small selection of coding languages. This paper could possibly be used as guidelines when implementing assets. How useful this information may be is uncertain and this paper is to the best of our knowledge the only of its kind.

2.2.1.2.5 Product derivation

The academic interest in product derivation for SPL is scarce. This fact was stated by [DEE05] and has according to our findings, or rather the lack thereof, not changed. Publications on the subject seams to be limited to on one hand generic concepts or practices as in [DEE05] and on the other hand very specific, often formal, practices intended for automation such as [SIE08]. The problem with the later is that they do not consider how the method fit into a more complete methodology or framework for SPL, they need a very well defined and stable domain and, most importantly, they lack in validation. A noteworthy exception to this is PuLSE-I which defines a process workflow for the product derivation. PuLSE-I is described in [Appendix D: The PuLSE methodology].

The derivation of products form SPL assets can be done manually by linking assets with "glue" code or the process can be automated provided that the preceding practices has a certain degree of formalism. Although a fully automated derivation process would be something to strive for, it can be very hard to achieve in practice. The most common practice found in industry would logically assumed be found somewhere in between these to extremes, an automated derivation of a common stub or skeleton, possibly adding variable features, while more detailed customizations are made manually. This assumption is however hard to back up with facts since this area of SPL development is barely mentioned in industry case studies.

2.2.1.2.6 Quality assurance and testing

Testing is an essential practice of SPL engineering. As assets are reused and incorporated into many products the consequences of an insufficiently tested asset is multiplied [GAN07]. In theory it seams easy. As each reused asset should already be tested in the previous project it was used in, it could be considered as functioning correctly for the new project unless the asset was changed in between. The authors of [GAN07] call this strategy Infrastructure-Focused Quality Assurance. They also state that the generalization of SPL assets and the multitude of ways they can be used in makes rigorous testing of these assets very problematic and often economically infeasible.

The other strategy is by [GAN07] identified as Product-Focused Quality Assurance, in which each asset is tested for every product. Two problems are identified with this strategy. Firstly, the assets are tested late and within the context of the current product, with the possibly of faults being mitigated to other products before detection. This may increase the cost of correcting the faults. Secondly, as the assets are reused in many products a lot of resources will be wasted on redundant testing.

These two strategies are described as extremes with many compromising strategies in between [GAN07], however they capture the problem of SPL testing in a nutshell. Few specific practices for SPL testing has been proposed as traditional object oriented methods are used [KAU03]. The SPL specific problem is a matter of scale.

According to [KAU03] published in 2003, SPL specific testing methods are immature but new methods was under research in both SEI and CAFÉ related projects. We have not been able to find any results of this research.

As SPL assets are expected to change over time, change management is a required practice to ensure continued quality [SUN08, SEI10]. Most frameworks for SPL include an explicit practice for change control and management. One such change management system is described in [SUN08]. The described process includes the categorization of change request according to the depth of the change, impact analysis of the requested change and validation that the implementation of the change solves the problem.

2.2.1.2.7 Discussion

The research field of software product lines have incurred interest from both academics and industry. And so is the fact in area of methods as well. Academics have proposed a rather vast amount of methods and theories that according to their own claims would solve one problem or another, although it is very hard to find validation of these claims. Many things have been tried in industry, there are many experience reports to support this. Unfortunately these experience reports does not reference or describe the methods and processes used in enough detail to make them repeatable.

[KHU09] concluded in their systematic review of domain analysis solutions that evidence to support claims of validity as well as usability and usefulness are lacking. Unfortunately we have to agree with their conclusion and add that this is the case for all research on solutions for SPL development, not only the domain analysis. This lack of validation makes it hard to make recommendations on what methods to use. No authors of previous reviews provided such recommendations. Most authors recommend their own methods but there seems to be no critical evaluation by a third party on any of the methods that result in a new recommendation.

In our opinion SPL development is more about the work processes then a collection of individual tasks represented by methods. The methods are tools to be used in the process, and tools come in different shapes and sizes to target a specific need. What works in one situation might not be suitable for the next, thus the choices in methods can only be left to the individual organization. However, our recommendation is to first concentrate on what to do rather then how. We know for example that feature modeling serves its purpose and works according to experience reports, but since the validation of the individual feature modeling methods is scarce and the experience reports do not mention which model they used one can not say which one works best in a particular context.

PuLSE (Product Line Software Engineering) methodology is one of few methods that we have found that considers the whole chain of SPL development. It is also one of the most well documented and validated methods within the field. Methods focusing on the processes, such as the PuLSE methodology, have a greater applicability as it considers each task as a link in the development chain and concentrates on how the links fits together rather than on the links themselves. To use something like the PuLSE methodology as a whole would be a more or less ideal SPL approach as it incorporates all the important aspects of SPL development. The experience reports using scaled down PuLSE processes have shown good results in industry and could possibly be considered one of the best practices for SPL engineering to date [SCH00, KNA00]. Summary of the PuLSE methodology can be found in [Appendix D: The PuLSE methodology].

2.2.2 Maturity of SPL and Organization

This section is divided into two parts. First there is a summary and discussion of research on SPL maturity. This is then followed by a discussion of organization maturity represented by CMMI in a SPL context. As explained in the introduction of this paper the decision to use CMMI was threefold. The acceptance for CMMI, its origin and the fact that ISO-TickIT that was also a candidate is undergoing a reconstruction.

2.2.2.1 Software Product Line Maturity

It can be hard to specify SPL maturity because of the fact that there are many diverse practices and one is not necessarily better than the other. Most papers discussing SPL maturity focus on how the commonality/variability engineering is performed and how well an architecture is enforced, [BOS02] being the most notorious of these papers. An exception to this trend is the BAPO framework where the practices mentioned could be considered as part of the architecture quarter of the framework [LIN04]. The other three quarters of the BAPO frameworks focuses on the business, process and organization aspects of SPL development [LIN04]. We agree with the BAPO on the fact that a SPL initiative should have its foundation in strategic business decisions and have an organization behind it who support and agrees with the initiative. However, even though we may touch on the organization area, our scope for this thesis is more focused on the architecture and process areas of BAPO and so we will assume that the initiative for SPL development is founded on a sound business case. As process maturity is covered by CMMI in this thesis this part will mainly focus on works related to domain engineering and architecture enforcement.

One of the earliest works on SPL maturity was presented by Jan Bosch. In [BOS02] he presents a maturity representation with five main steps and two additional

steps that are put somewhat in parallel to the main maturity steps. These two additional steps are explained as practices that are normally used when an SPL grows large. In addition the overall maturity representation, the maturity of artifacts are also covered and related to the overall maturity of the SPL.

In this early paper the different maturity steps are described and given an example although they lack a good definition. This is somewhat corrected in [BOS03]. The maturities are given a more structured definition and the concept of multi-scope SPL organizations are introduced. A multi-scope organization is according to [BOS03] an organization where different levels of SPL maturity has been reached at different levels of scope. For example, an organization can has a Scope 1 which is a common platform level and inside this they can have several sub scopes. Scope 1.1 can be a regular SPL and scope 1.2 can be a so called product population which is a set of products created by assets without a PLA. The assets from Scope 1 are shared in both these, but assets inside the sub scopes are not shared. Additional sub scopes can exists, a specific product inside the SPL can be a configurable product base. Containing products that can be several different products when installed and perhaps additional products after runtime configuration.

The work on classifying SPL maturity continues in [JAR04] where introduction and binding phases of decisions are considered to define the maturity of the practices. The idea presented is that the further you can go into development without making any permanent decisions the higher the maturity of the practice is. This is rather reasonable since it is well known that requirement and other circumstances often change during the development life cycle which forces decision changes to be made. If the practice used is mature by this definition such changes can be made without any costly rework.

[NED07] presents an investigation on how 13 companies move between the maturity levels defined by [BOS02] as their business changes and practices evolve. The paper also records a number of organizational attributes such as products, size, goal, drivers and methods of transition. The study resulted not only in a mapping on how and why companies can move between maturity levels but also identified a number of new states beyond the maturities defined by [BOS02]. One reason for this may be that the study include a larger number and a wider range of companies than any of the previously mentioned papers. The new set of states may however not be very good as a maturity representation as the number of states and the ways to move between them would make such a maturity model quite complex.

All the above mentioned papers are based on the same original concept. All we have mentioned above, except from [NED07] are also authored or co-authored by Jan Bosch. The concept has also been adopted by the BAPO framework [LIN04] coauthored by Jan Bosch. The fact that one author is behind all of the prominent works could raise some concerns. Especially since no other views on SPL maturity seems to have been presented to challenge the established concept. We do not suggest that the established concept proposed by Bosch is flawed or inadequate. We just want to point out that there is no rivaling view to compare it to. There is also yet to come a paper or experience report that show that this model is in truth useful as a maturity model. That aside, we are skeptical to whether what Bosch suggests is really a maturity scale. There is very little to confirm that a Configurable Product Base approach presented as the highest SPL maturity is in any way a better approach than what Bosch defines as a Software Product Line. There are situations where a Software Product Line would be preferable to a Configurable Product Base. As the term maturity is inherently a

positive term a higher maturity should always indicate a better practice than a lower maturity.

2.2.2.2 SPL and CMMI

One of the research questions of this thesis was whether an SPL initiative and CMMI initiative could benefit from each other, either by increasing the organizations process maturity by introducing SPL practices or by making the SPL approach easier by having a high process maturity within the organization. To answer this question we looked at the concepts behind the two initiatives and tried to find similarities. Our literature review also found a few earlier works asking similar questions. Starting with related work and concludes with our own investigation. As basis for this investigation we used the related work and there conclusions and results as well as [Appendix G: CMMI and SPL Summary]. This appendix contains a CMMI background and a list with CMMI process areas and our investigation on their importance in SPL.

We studied the concepts of SPL engineering and compared them to CMMI's process areas with the intent of finding out how well the CMMI Process Areas (PA) could be fulfilled just by implementing SPL practices.

The motivation behind this research is that SPL needs more discipline, coordination and a more unified way of working than regular development. It also has higher dependencies as well as needs higher predictability and quality [JON05]. CMMI has been documented to provide this. Another motivation for this research is that there exist many process improvement initiatives that is considering SPL adoption as well. Companies in this situation have often asked if and how they can combine and reconcile these two major change initiatives [JON05].

2.2.2.3 Related work

Jones and Northrop [JON05] have considered the issue of reconciling a CMMI and a SPL initiative and investigates the SPL adoption process in organizations which uses CMMI models as their process improvement method. They show how certain CMMI process areas can be used as a foundation for SPL practices.

Fredy Navarrete et al. performed a initial analysis on combining CMM models and Agile Methods in an SPL environment [NAV06]. They balanced these two software engineering contexts and took advantage of both approaches. CMMI specifies what to consider when defining the disciplined process, while Agile Methods specifies how to address the different software practices in the SPL. When discussing CMMI in a SPL context they refer to the comparison of practice areas and process areas in [JON02] and [CLE02]. When investigating how well Agile methods worked in a SPL context they found practices which had positive effects and some which will have negative effects and some that has both. They conclude that SPL development requires coordination, discipline, communication and shared knowledge. One way of getting this is to adopt CMMI for the process discipline and to coordinate your processes, and use Agile Methods to support communication and shared knowledge. In short, use CMMI to define what to do and Agile Methods to describe how you do it.

There have been some comparisons between CMMI and SPL practices, especially the Framework of Software Product Line Practices [NOR07]. This is probably since both are created by SEI, and thus there are more likely to be some similarities when comparing these. Clements and Northrop [CLE02] are two of the researchers on SEI and they state that if an organization do not have a strong process culture then the SPL

adoption process will be tough. They also state that processes that are well defined includes boundaries and responsibilities for each employee for increased collaboration and cooperation which is definitely a need for SPL development.

Another comparison is made by Jones and Soule [JON02]. They first perform a general comparison on focus, coverage, levels and if they contains how to information. All these differ between SPL and CMMI. Then they perform a more detailed comparison of the CMMI process areas and the frameworks practice areas. They list all practice areas of the SEI Framework, then they list which if any process areas of CMMI that cover that practice area and how strong the relation is. Most of them have some relation, few have a stronger relation between them like Technical Risk Management/Risk Management, Training/Organizational Training and Configuration Management/Configuration Management. They also identifies some practice areas that do not have any representation in process areas like Understanding Relevant Domains, Mining Existing Assets, Launching and Institutionalizing and Scoping.

On the contrary they also state some that do not correspond to any SPL practices, but we say that this does not mean that they are unnecessary in SPL development. These are Process and Product Quality Assurance, Organizational Process Focus, Organizational Process Performance, Quantitative Project Management, Causal Analysis and Resolution, and Organizational Innovation and Deployment. These are in fact very important for SPL engineering although they are not included as specific practice areas. Lets take Process and Product Quality Assurance as an example, we think this is important or even more important for SPL then for regular development. Low processes quality leads to corrosion of SPL assets and as the assets are reused multiple times a problem with product quality will multiply if not handled. [JON02] also refer to two case studies which have introduced SPL and process improvement together, not only do they show that these work well together but also indicates a increase in benefits when using these in cohort with each other.

In a test case performed by Tara Regan and Donald Reifer [RAG98], they incorporated SPL into an early version of CMM, the SA-CMM. This resulted in a set of changes proposed to the SA-CMM model to in corporate the SPL practices. These changes were then included and some pilot projects were executed with good results. Their effort showed increased reuse and that a incorporation between the two initiative should be possible if you are willing to change some minor things.

In an initiative by the European Software Institute (ESI), Nokia and Siemens [KÄN05] they investigate the PA's of CMMI as well. They create the Family Maturity Framework (FEF) which defines maturity level of the four BAPO dimensions (Business, Architecture, Process and Organization). The article describes a few general thoughts on each maturity level and its role in a CMMI context. Also provided are a discussion on each PA and specific parts of PA's and what to consider in a SPL environment. Their study is therefor similar to the one we make in the following section. The reason for this is that we we planned our comparison we had a different focus. Our focus was to discover how CMMI maturity levels affect the SPL process and since we did not find such a connection the comparisons turned out similar. They were in fact practically the same with a few exceptions and their result concurs with our results.

2.2.2.4 Discussion

One similarity between SPL and CMMI at a quick glance is that both have a strong process focus. SPL development is a long-term investment and requires a

certain amount of development process enforcement in order to ensure the SPL integrity over time. CMMI is all about defining, maintaining and improving the organizations processes.

At first it looked like quite a few of the CMMI PAs could be fulfilled with ease by SPL processes but there was always something that was missing. The SPL processes answers the question "what should be done?", and sometimes states "this is how we do it". While these SPL statements often was direct implementations of CMMI goals and practices, other goals and practices was left ignored. Practices concerning evaluation and improvement of the processes was rarely fulfilled by the SPL processes.

When investigating further on the correlation between the two areas we started looking on each of the practice areas and their importance in SPL development [Appendix G: CMMI and SPL Summary]. We found that most are equally important in SPL as in regular development but some are extra important for SPL. These are Configuration Management, Process and Product Quality Assurance, Organizational Training, Product Integration, Requirements Development, Technical Solution, Validation, Verification and Causal Analysis and Resolution. We found none that are unimportant for SPL, see Table 9 for summarized importance and complexity, Table 10 for definitions and [Appendix G: CMMI and SPL Summary] for a more detailed discussion. There also are some SPL areas that are not represented by the Process Areas at all, such as asset development and domain analysis.

Table 9: CMMI process areas, their importance and complexity in SPL

Process Area	**SPL Importance**		**Process Area**	**SPL Importance**		**Process Area**	**SPL Importance**
ML2 – CM	++!		ML3 - IPM	+!		ML3 - TS	0
ML2 – MA	0		ML3 - OPD	0		ML3 - VAL	++!
ML2 – PMC	0		ML3 - OPF	0		ML3 - VER	!
ML2 – PP	*		ML3 - OT	+		ML4 - OPP	0
ML2 – PPQA	++		ML3 - PI	++		ML4 - QPM	0
ML2 – SAM	*		ML3 - RD	+!		ML5 - CAR	++
ML2 - REQM	+!		ML3 - RSKM	0		ML5 - OID	0
ML3 - DAR	++						

Legend: (++) Crucial, (+) Important, (0) Equally important, (-) Less important, (–) Unimportant, (*) Important in certain cases, (!) Extra difficult.

Table 10: Definitions of importance

(--) Unimportant	This PA lack relevance to SPL development.
(-) Less important	This PA is not as well utilized in SPL development or there are other SPL related processes that work around this PA.
(0) Equally important	The baseline. We could not find any differences with this PA when having SPL development processes compared to

	traditional development.
(+) Important	SPL development practices utilize this PA a lot. Putting extra effort at this PA may increase the effectiveness of the SPL.
(++) Crucial	SPL development practices rely on this PA. Poor execution of this PA may cause the SPL to fail.
(*) Important in certain cases	This PAs importance depends on what SPL development methods and techniques are used. Otherwise it is equal to (+).
(!) Extra difficult	When using SPL development you will have to take this PA one step further regarding analysis or execution.

To initiate SPL, a software organization needs needs to have Defined Processes at least. First reason for this is that then there exists established guidelines on how to create the processes to avoid unnecessary inconsistencies, which can be devastating for SPL development. Second reason for having Defined Processes is that there is a process asset library available, much like the asset library of SPL. Viewing the processes as SPL assets is not a completely new idea. Another advantage of Defined Processes is that the processes are general and the same over all projects; this also prevents inconsistencies as well as simplifies coordination of processes.

On the other hand, having lower process maturity will not prevent the adoption of SPL practices, but it will probably complicate the process. It could cause issues with ambiguity, inconsistency, lack of communication and coordination. Having higher process maturity should simplify the adoption process further. In Quantitatively Managed Processes there is an increase of predictability and increase of capability understanding. In Optimized Processes there is incremental improvement and a significant decrease of defects and other issues, which would also simplify the transition. Also we could not find any reason not to have a CMMI initiative in a SPL development organization.

2.2.3 The SPL Transition

The transition towards a SPL approach is not to be taken without considering all ramifications it brings. It is potentiality a huge organizational change and full knowledge of the organization's situation and current process is needed in order to choose the adoption approach most suited for the organization.

This section has the following structure: first there is a section describing the terminology used in works on adoption approaches. After that follows a section on the definition of three different types of approaches. Finally the section is concluded with a section discussing SPL adoption in general.

2.2.3.1 Adoption terminology

When going through the different methods and case studies on how to adopt SPL we identified that there is no agreed terminology. Some use the same terminology but have different definitions of the words used, but it is more common that different terminology is used for to explain the same phenomena. To clarify we compiled a

glossary of this terminology. We have included at least one definition made by an author of a adoption related work and added our own comments on how the term is normally used in the literature.

Proactive

Definition 1: The organization analyze and designs the complete software product line. Based on the analysis and the design the assets are created [KRU02a].

Definition 2: Analyze, architect, design and implement all product variations and commonalities based on predictable requirements [KRU02b].

Comment: Used to describe an approach where the user tries to predict future changes in requirements and incorporate them in the design of the system. This is time consuming and very hard but sometimes preferable since it may be even harder to change the design at a later date. Most approaches for SPL are proactive to one degree or another.

Reactive:

Definition 1: The organization incrementally expands their product line as new demands arises [KRU02a].

Definition 2: Implement several variations for each development cycle based on predictions [KRU02b].

Comment: In contrast to a proactive approach a reactive approach does not put as much effort into predicting future changes but incorporate a flexibility to allow changes to happen and then react to them. Reactive elements are usually incorporated into most approaches since it is practically impossible to predict all changes that may come in the future.

Extractive

Definition 1: The organization extracts the assets from existing software systems [KRU02a].

Definition 2: Make use of existing software products as the base for a new product line [KRU02b].

Comment: An extractive approach is not focused on how the SPL and its creation is planned as proactive and reactive approaches but focuses on how the asset creation is performed. Thus an approach can be proactive/reactive and extractive at the same time. The opposite of an extractive approach would be an approach where all assets are developed from scratch, although there are no term used for such an approach.

Revolutionary

Definition 1: PLA and components are developed based on current requirements and future predicted requirements [BOS02].

Definition 2: The development of a complete PLA and core assets before developing the first product in a new domain. [MAT02]

Comment: The revolutionary approach to SPL adoption embraces the idea to start from scratch and do everything correct from the start. Everything is planed and then executed. First when all assets are created and everything else is in place the product production is started. A revolutionary approach would by default be a proactive approach. Another word often used to describe a revolutionary approach is Big Bang.

Evolutionary

Definition: PLA and components evolve with requirements [BOS02].

Comment: Evolutionary is often used to describe an approach that iterates the creation of assets with product development. Most approaches that are used in industry are in some sense evolutionary. An evolutionary approach is often based on one of two facts. It is hard to predict changes and to create everything first will take a long time and postpone the product release schedule. Sometimes the word incremental is used with the same meaning but usually the word incremental has to do with the organizations transition while the word evolutionary focuses on how the assets and products are developed.

Incremental approach/transition

Definition: Limited and incremental investments is given to the SPL initiative while the major part remains in traditional product development [BÖC02]

Comment: Starting with a single department changing to the SPL approach and if and when they are successful, you may add other departments incrementally. May also refer to how the SPL artifacts are created (see evolutionary). Piloting is a special form of incremental approach. An incremental approach is often advocated with the phrase incremental return of incremental investment.

Piloting

Definition: The SPL initiative is started by creating a pilot which is either one project or a similar product that some processes, activities or results that can be used as a basis for the future SPL to be [BÖC02].

Comment: Piloting can be done in several different ways. A new product can be created with product line methods and the assets of this product becomes the first assets of the product line. A different type of pilot is to develop a toy product or prototype, meaning you create a product that is significantly close to your real product but it requires less money and time. This way assets can be identified and sometimes extracted.

Applying SPL on a pilot is obviously less risky and a lot can be learned from this for when to use it in full scale development. Also see incremental transition.

Big Bang

Definition: The SPL is created and adopted immediately [BÖC02].

Comment: In a SPL transition approach context Big Bang is considered more or less a synonym for a revolutionary approach. Sometimes it may have the meaning of a strategy where the SPL approach is adopted and enforced immediately without trying it out first.

Opportunistic reuse

Definition: The opportunistic approach relies on the developer to select and combine the correct pieces of software that fit the current problem [BOS00].

Comment: An approach that is characterized as opportunistic does not plan for reuse. When a peace of code is found reusable it is remade into a reusable asset. Reactive approaches is more likely to utilize opportunistic reuse. Some evolutionary approaches incorporate opportunistic reuse, although opportunistic reuse is usually looked down upon by researchers.

Planned reuse

Definition: This approach requires that the whole organization emphasize the development of reusable artifacts with correct abstraction and variability for the products developed by the organization. [BOS00]

Comment: Planned reuse is what is desired in SPL engineering. Reusable assets are identified by analyzing the domain and previous products before they are developed. Proactive approaches utilize planned reuse. Revolutionary approaches take the planed reuse to the extreme while evolutionary approaches try to plan as much as possible.

2.2.3.2 Adoption approach types

In this thesis, we characterize three different types of approaches to initialize an SPL based on previous studies [BÖC02, BOS00, KRU02a, KRU02b] and the terminology in Section 3.3.1. All proposed types and specific approaches fall in at least one of our defined types. Some of the approaches may fall within two of the types depending on how the activities they specify are planned and executed. This is further explained in Section 4.3.

The reason why we made this classification was to have a starting point in the process to choose an approach. We will return to these approach types later in the guidelines section of this paper.

Incremental with Iterative Analysis: This approach is a reactive and evolutionary approach, characterized by systematic but opportunistic reuse. The company has taken the business decision to advocate reuse and develop a reusable asset library. There is no overall plan for a SPL and if there is a reference architecture it is usually not enforced and can be modified as necessarily. The development is more oriented towards single system development but commonality and variability analysis in a domain context is carried out in each project to look for subsystems or components that could be made reusable.

Incremental with Pre-analysis: This approach is also evolutionary but is in contrast to the above mentioned approach proactive and utilizes planned reuse. The approach is incremental as the company tries to have a stepwise transition from single system development to SPL development. The domain and the products are analyzed first and reusable assets are identified. The assets are then designed at least on an abstract level to ensure that they all fit together at a later stage. The development can then still be rather focuses on product development for a time while more and more reusable assets are developed with each product. A development plan is often created that specifies which assets are to be developed in which order and within which project. With each new project the development will shift slightly more toward product line development.

Big Bang: The Big Bang approach is proactive, revolutionary and very sequential. The domain and products are analyzed, reusable assets are identified, the system is designed, assets are created and products are developed with the assets. If the company is new to SPL it would be a huge leap of faith to start out with this approach. It is true that there are many long term advantages with this type of approach. The SPL will likely be easier to maintain and product production will be less time consuming. But it requires a large up front investment and since a considerable amount of time will be committed to developing assets the product release schedule will have to be put on

hold. Early theoretical works on SPL is of this type and some case studies conducted in military projects, but most theoretical works has abandoned this approach as it is rarely used in industry. It is to big of a risk and to large an investment.

2.2.3.3 Discussion

2.2.3.3.1 Readiness

Before initiating a SPL adoption, the company need to make sure that its organization is ready for the change. To investigate on this there are several readiness analysis methods. A summary of them can be found in [Appendix E: Readiness Tools].

When comparing these different readiness analysis methods, there are some differences as well as many similarities. One difference is the effort involved, but the effort is almost equivalent to the amount of information gathered. The one that requires the least effort is the Product Line Potential Analysis that is intentionally designed to be fast and easy to use. After a quick analysis this method will provide recommendations on how to proceed. Product Line Potential Analysis does not analyze benefits and risks like the Domain Potential Analysis [SAG00, BAN00] and the Product Line Benefit and Risk Assessment [SCH02a] and neither does it consider the required expertise like the Technical Probe [SEI10b] and the Knowledge Assessment [BÜH04]. Since it focus only on the question, "Should we go ahead?" it can be done in matter of hours. This is a significant benefit since it is likely that management would want to keep cost and effort at a minimum, especially before they know if transition to SPL will generate profit or not.

If an organization wants a more thorough tool there exist several alternatives, first one of them is the Domain Potential Analysis. This gives a good overview of the different risks and the different economic gains in a certain domain. The Product Line Benefit and Risk Assessment is similar but not as thorough and should suit for situations where you want a simple overview of risks and benefits and not want to spend so much time and effort. The most extensive and thus time consuming is the Product Line Technical Probe which is tightly based on the SEI Framework which makes this a desired approach if there is available time and money for it. Another tool that uses the SEI Framework is the Knowledge Assessment, this gives readiness of separate areas and enables improvement on eventual weak areas.

The methods have some similarities in what information to gather to be able to make the decision about SPL initiation. All methods identifies current status and potential of one or more of the following areas, the market, the organization, the business units, the individuals, the architecture and the assets. The method of how to do this are also almost unified, utilizing interviews and questionnaires with stakeholders. They all compare benefits and risks against each other in some way, whether they call it strengths versus challenges or inclusion versus exclusion criteria has less affect on the result. A couple of methods uses a SPL framework as a reference. This makes them better suited when aiming to follow one of those frameworks.

When investigating readiness some prerequisites was found, for once, all researchers agree on that to succeed in the transition you need to do proper preparations. This is also seen in some case studies. In one of these case studies, by Kircher et al. [KIR06], they learned that the organization should attain a certain maturity level before initiating a SPL approach.

They also state an additional four prerequisites. First, the requirement should be reasonable, this is to be able to perform a commonality and variability analysis. Second, the architecture should be clear and well described to be able to realize the identified variations in actual design. Their third prerequisite is to have automated system tests in order to enable fast adaptations. And finally to have advanced configuration management, without this every change could lead to inconsistencies. In an article on Agile SPL by Hanssen and Fægri [HAN08] they state that the most fundamental precondition for SPL is to have an accurately predicted market. Not having this may lead to that the SPL initiation consumes resources instead of increasing and thus slowing down the software production.

2.2.3.3.2 Adoption approach

When choosing a specific adoption approach there are plenty of alternatives, those we found the best are described in [Appendix F: Adoption Approaches]. Defining current state and desired state will simplify the choices around the transition since this provides a clear picture on what and where changes are needed in the organization. Another way of figuring out where changes are needed is to use a framework and evaluate how well the organization handles each practice area.

Some of the adoption approaches uses the SEI framework for this and it should be a desired choice as it incorporate all essential areas of SPL. The focus of most approaches are to lower the initial cost, to simplify the transition by taking it in small steps or to help in making the correct decision. Introducing SPL is a big organizational change and requires much funding, committed and knowledgeable management and will most likely be hard and confusing at some point in the transition. These problems are well recognized among researchers and practitioners.

There are more variety in the approaches suggested by researchers then the ones used in industry. A reason for that could be that researchers are trying to improve existing approaches and creating new approaches to tackle unsatisfactory solved issues while companies want to use a stable and previously used approach. There are some that could be considered standard approaches [SIM02, CLE06, KRU02a, ZHA06]. Some put more or less weight on certain activities and a few of them continues after analysis and design with activities such as implementation, integration and testing. Revolutionary approaches are scarce, one of the few existing are QADA [MAT02] which is very extensive and has a strong architecture focus. There are some case studies that explains revolutionary approaches but these are almost exclusively in the military where funding is not a issue. Problem with these though are the descriptions of their way of work which are poor or none in many cases, one exception of this is shown in [KOS06] which contains their practices and some description of them. Table 11 shows the main characteristics of the adoption approaches.

Table 11: Adoption Approaches Comparison

Adoption Approach	Source	Derives from	Validation	Repeatability	Novel	Adoption Type
A knowledge based transition approach	[MAT05]	Research	None	Medium Low	Yes	Evolutionary
A revolutionary initiation approach	[MAT02]	Research	Hypothetical Example	High	Yes	Revolutionary
Adoption Factory Pattern	[CLE06]	Research	Hypothetical Example	Medium High	Yes	Several Types
An adopting and institutionalizing approach	[BOC02]	Research	None	Medium High	Yes	Several Types
An evolutionary lightweight iterative adoption process	[SIM02]	Research	Hypothetical Example	Medium High	Yes	Evolutionary
BigLever's Software's transition approach	[KRU02a]	Research	Hypothetical Example	Medium High	No	Several Types
Case Study at Bosch Gasoline Systems	[STE04]	Industry	Industry Case Study	Medium Low	No	Evolutionary
Case Study at Engenio	[HET06]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study at HomeAway	[KRU08]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study at Overwatch Textron Systems	[JEN07]	Industry	Industry Case Study	Medium	No	Evolutionary
Case Study at Siemens AB	[KIR06]	Industry	Industry Case Study	Medium	No	Evolutionary
Case Study in Car Periphery Supervision Systems	[THI01]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study in Small Embedded Systems	[STO02]	Industry	Industry Case Study	Medium Low	No	Evolutionary
Case Study in the Digital Audio & Video Domain	[KIM07a]	Industry	Industry Case Study	High	No	Evolutionary
Case Study on systems for automatic guided vehicles	[SVA02]	Industry	Industry Case Study	Medium High	No	Evolutionary
Enhanced Unified Process for Product Line	[ZHA06]	Research	Industry Case Study	Medium High	Yes	Evolutionary
Military Case Study the JTRS program	[KOS06]	Industry	Industry Case Study	Medium Low	No	Revolutionary
Observations from Multiple Case Studies	[PAT04]	Industry	Industry Case Study	Medium	No	Evolutionary
Staged adoption of software product families	[BOS05]	Research	Low	Medium	Yes	Evolutionary

Derives from: *States whether the approach comes from researchers or from industry.*

Validation: *States the level of validation of approach. Low are when the approach is based on previous experience, hypothetical example is small internal made up case study and industry case study are when the approach has been used in industry and thus gained validation.*

Repeatability: *Defines how good the approach can be repeated and used again. Thorough descriptions, illustrations and examples increases the repeatability.*

Novel: *This is a yes or no question on how novel the approach is, novel is not always a good thing.*

Adoption type: *Simply states what adoption type it belongs to. Can basically be evolutionary or revolutionary, some defines or can be used for multiple types though.*

The transition to SPL is difficult to say at least, and thus there are many novel ways of tackling it. One way is to consider it as a knowledge management issue and focus on knowledge gaps [MAT05], another is to consider certain decision dimensions [BOS05]. Instead of creating novel approaches there are some that tries to simplify the procedure instead by providing road maps or easy overviews. Such roadmaps is shown in [BOC02] and [CLE06].

The articles presenting approaches used in industry differ in the sense that they are more focused on the results than on how the transition was made. Most of them uses a standard well used approach comprising of domain modeling, requirement gathering, feature analysis and architecture creation [THI01, KIM07a, STO02, KIR06, STE04.] Almost all case studies presents lessons learned, challenges or recommendations, these could be read to enlighten management on possible future issues or challenges that could arise.

There is a noticeable difference between adoption approaches that researches suggest and what is actually used in industry. Those suggested by researchers often describes the way that is theoretically the best way of working but which might not work in industry. Reasons for those approaches to not work in industry could be lack of funds, time, staff or experience. Approaches developed and used in industry are often rather simple in comparison to the academic approaches. They are naturally created specifically for the organization that create it and unfortunately they are not always very well described. This makes them hard to adopt by another organization.

2.2.4 Guidelines for the SPL Transition Process

In this section we will present our suggestion on how to prepare and initiate a transition to SPL development. The majority of the guidelines are constructed by combining previous works and by putting previous works in context of each other. How the works was put together was based on the previously made analysis of these works and what their purpose and aims are in combination with analysis of case studies and experience reports that report on how transitions are made in industry.

The guidelines are constructed to fit all types of organizations. Depending on the attributes of the organization there exists different roads to take, attributes such as company size, type and number of products and development process. In this section we describe organization specific guidelines depending on those attributes.

When looking at the size, in general a smaller organization has access to less funds and should therefor be more careful in the transition. We recommend a incremental approach, with a thorough readiness control to make sure that the organization will handle the process. In organizations with more than one department there could be a rather safe approach to introduce SPL in one separate department referred to in [BÖC02] as incremental introduction. The other departments can continue with regular development and the knowledge gained in that department can act as basis for other departments to follow. If there are some employees with experience of SPL they could be moved to the separate department. Those employees working in the separate SPL department could after the introduction is complete, be viewed as experts. These experts could then be placed in other department to guide and lead the SPL adoption process. A challenge in a large organization is the distribution of knowledge, therefor focus must be on the communication to fully convey the goals and objectives to all parts of the organization and to all personnel. Otherwise there might be lack of commitment and knowledge amongst them.

When discussing type of products and the amount of products that the organization currently develops, it is important to have several similar products, otherwise the SPL benefits will dissipate. Beside that the type and number of products should not affect the choice of approach.

The development process i.e. bespoke or market driven, will very much affect the SPL. The market driven process is more suited for SPL because there is more control on what to provide and develop. It is easier to plan and the products to create can be fitted perfectly in to the PLA. If a new requirement arise on the market, then this can be planned in when suitable and not necessarily on the next launch. This is not the case in bespoke development, customers have large influence on the development and can come with late requirements to the product demanding functionality that is not provided by the SPL. Then the organization must adapt to this and either extend the SPL or make some product specific adjustments. For a bespoke organization we recommend to have the core of the system as a SPL and for the rest of the system be as flexible as possible to adjust the product in relation to the customer.

The guidelines is structured sequentially as follows:

1) Readiness Control
2) Choice of Adoption Type
3) Choice of Specific Adoption Approach.
4) Choice of Framework
5) Choice of Methods
6) Additional Development Guidelines

The need for readiness control and preparation is advocated by several researchers. There are several tools and methods with different starting points and aims for this purpose. The first step in our guidelines combine a few of these tools and methods with a result that should give a solid foundation for the transition. Most of the privious analysis made for this is presented in Section 2.2.3.3.1.

The following two steps aims at specifying how the transition is to be conducted. These steps are mainly based on our analysis of adoption approaches made in Section 2.2.3 with the addition of variables to consider when choosing approach. The variables have been extracted by analyzing the differences between adoption approaches and how they have been used in case studies.

Step four, five and six are all based on discussion and analysis performed in previous sections as these steps gives guidance and suggestions on the choice of suitable frameworks and methods. These are very generic as the validation of these areas are generally very low. The discussion that are the basis for step four can be found in Section 2.2.1.1 and more information and descriptions of frameworks can be found in Appendix B. Similarly the discussion for step five can be found in Section 2.2.1.2 and more information and descriptions of methods can be found in Appendix C and Appendix D.

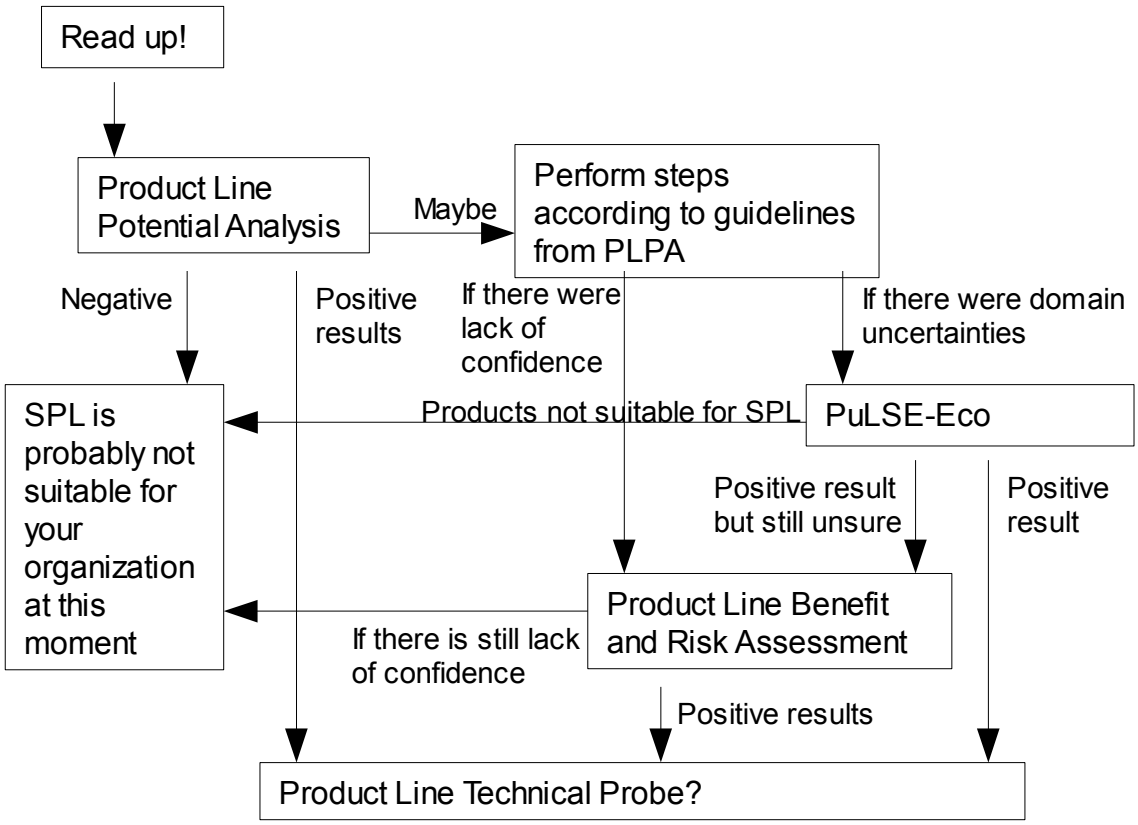
2.2.4.1 Readiness Control

2.2.4.1.1 Initial investigation

When an organization first starts to consider a SPL approach it could be overwhelming, thus we suggest to read up on the subject to begin with. One perfect

way to do this is reading the Software Product Line Overview [NOR07] and surrounding areas on the SEI homepage and take a quick glance at the SEI framework to discover major differences from regular development. There could also be good to check out the four BAPO dimensions, Business, Architecture, Process and Organization. These are the corner stones of the Family Evaluation Framework which evaluates all four dimensions separately. This will give the organization further insight on what is needed for SPL.

Figure 5: Readiness Control



When there is a basic understanding on SPL it is time to discover how ready the organization is towards the adoption. In our literature review we found several proposed tools for this. We recommend to start with the Product Line Potential Analysis [FRI04], which is a fast tool [FRI04] that answers the question "Should we go ahead?". This is a good start that will guide the next steps. If there is uncertainties about the domain we recommend performing a scoping analysis like the PuLSE-Eco which answers if the products are similar and suitable for a SPL. If the organization needs more confidence and commitment after the PLPA or PuLSE-Eco we recommend the Product Line Benefit and Risk Assessment [SCH02a] or the Domain Potential Analysis [BAN00, SAG00] which more go into detail on risks and benefits. Both these answers the question "Does the benefits overcome the risks?" When there is enough commitment we recommend the Product Line Technical Probe which is more of a preparation tool than a readiness control. The Technical Probe is a bit time consuming but provides a solid foundation for the SPL process. It identifies weak areas to work on

and strong areas to make use of. If any of the readiness tools provide negative results, then we recommend to consider if a SPL approach is the way to go. If the organization still wants to perform the transition after a negative results extreme care should be taken and an approach with less prerequisites and less risks should be used. All these steps are visualized in Figure 5.

If the Product Line Potential Analysis indicated a legacy architecture that could be reconstructed into a PLA then as an additional preparation for SPL the organization may want to perform an ATAM, or perhaps an EATAM [KIM08]. A reason for this is that a documented reason for successful adoption is to have a reference architecture ready before the transition starts [JEN07].

2.2.4.1.2 Preparation

After the initial investigation the organization should perform a Product Line Technical Probe [SEI10b]. The Probe is more of a preparation tool then a readiness control method and might be slightly time consuming. However it should be worth the effort to get a strong foundation for the SPL initiation process. The main part of the probe consists of interviews. To help with what questions to include in the interviews an article by Büher et al. [BÜH04] could be used. In this article they present questions to ask to gain sufficient information about the organization, business, development units and individuals to be able to choose the most appropriate way of initiating a SPL. Combining these should give an organization all the information needed to be able to choose approach, framework and methods for the SPL to be.

With help of the Probe, practice areas in the current development that can be considered strengths and practice areas that could be considered as weakness when transitioning to an SPL approach should have been identified. Weaknesses should be improved upon to gain a stable and efficient SPL engineering practice. They may also limit the choice in adoption approach and techniques to be used. It is equally important to make use of the strengths identified to make the transition process easier. Make use of the information gained through the probe in the following step.

2.2.4.2 Choice of Adoption Type

After the probe the organization should have a better understanding of how well prepared the organization are for a SPL initiative. The next step is to choose the type of approach to use. Earlier in this paper we characterized three types of approaches (see Section 2.2.3). When choosing an approach type there are a number of variables to consider, but in the end it all comes down to how much risk the company is taking and is willing to take with each approach. In the following subsection we describe the variables affecting the choice we have been able to identify from case studies and experience reports.

2.2.4.2.1 Variables to consider

Level of commitment to the initiative among developers and management.

Low commitment to the initiative among management may indicate that without immediate positive results the initiative can get abandoned. An approach with fast return of investment is recommended in this situation, even tho such an approach has less long term benefits. Low commitment among developers may be because of resistance to change in work procedures or a belief that the change will increase their work burden. The result of such resistance may be that the introduced procedures are

not carried out in practice, or that process discipline may severely drop as deadline approaches. In this situation the recommended transition approach would be a gentle and progressive approach that keeps as much as possible of already existing work procedures.

Initial and overall costs

The costs of a SPL initiative is high as it is a long term investment. In general you could say that the higher the initial cost in the transition approach is, the lower the overall cost will be. It also affects the amount of and time to return of investment. The reason for this is that with more initial funding a more thorough analysis can be conducted and a more complete set of assets be developed before product production is started. This should result in better and more complete assets, less asset development for each new product and less rework at a later time due to better asset design. It also means that the time before the first product produced from the SPL is launched will increase and with that the return of investment will be delayed. The return of investment is potentially greater when a larger initial investment is made, but there is always the risk that the initiative will fail or be abandoned before the initiative has paid of. For a company that do not have massive funds or previous experience with SPL or SPL like techniques, we recommend an approach with lower initial cost and faster but lower return of investment. This minimizes the risk of losing money on the initiative and if the initiative fails completely before it pays of, the amount of investment lost will be low.

SPL skill level and knowledge

To plan and execute a SPL initiative requires knowledge about the domain and skill in especially analytical techniques, but also in architecture and design areas. The more formal and potentially beneficial the approach is the more knowledge and skill is required. These skills can be learned as the transition progresses, but this would mean taking a higher risk. Training before the initiative would be preferable but requires a larger investment. A lightweight, easy to use and iterative approach is recommended if the company lacks in this area.

Domain stability

The stability of the domain hugely influences the amount of risk that are being taken when transitioning to a SPL approach. In a domain that changes frequently a large initial investment in a SPL initiative is not advisable. Incremental investments focusing on the areas of the domain that shows indications of stability however may later free resources that can be used to respond to changes in other areas, possibly giving the company a competitive edge.

2.2.4.2.2 Adoption Types

These adoption types where identified earlier in Section 2.2.3. In this section we extend the descriptions of the approach types with benefits and risks. Choosing an approach type helps to limit the number of adoption approaches to used.

Big Bang (revolutionary, proactive)

Characteristics: High risk; High commitment; Large initial funding; Highly trained staff; Very stable domain.

The big bang approach type is on top of the risk scale. The SPL is planned and designed ahead of the initiative and an a large portion of the assets are developed before product production is started. This means that the initial investment cost is rather high. Since it takes quite some time before the first product hits the market with this approach it is not advisable to chose this approach unless the market and domain is stable, the domain is well known and the staff and especially management is committed to see it initiative through. In the Big Bang approach the processes are in place correctly from the beginning which increases chance of longer life time of SPL and less corrosion, that is if the processes are followed. Another advantage of this approach is that it has larger potential profit and with a less total cost but in one big sum. It takes longer time before first return on investment but the cost per product is small once the SPL is up and running. The staff must have some knowledge of what they are doing and have a certain amount of process discipline, since it will take some time before the initiative have paid of and the failure or premature corrosion of the SPL would be very costly. This approach is not recommended unless the company knows what they are doing, what risk they are taking and is very committed to SPL engineering. This means that this approach is for those who already work with SPL and either recreate an existing SPL or create a new SPL to target a new market. When using this approach the aim will be set for what Bosch classifies as a Software Product Line maturity or above.

Incremental with pre-analysis (evolutionary, proactive)

Characteristics: Low to medium risk; Commitment among key staff and management; Low to medium initial funding; Champion with knowledge to plan and lead the work; Somewhat stable domain.

An incremental adoption approach with pre-analysis starts out in a similar way to the big bang approach. Domain analysis, architecture and design are conducted and executed prior to the start of asset development. But instead of developing the majority of assets first and then start the product development, the assets are developed more or less simultaneously with the products. For each development project a subsystem or a smaller collection of assets are developed for reuse. This would mean that in the first products released from the SPL there would be only a small portion of code that are developed as reusable assets, the rest of the product is developed as a traditional single system. But with every product released, more and more of the underlying code will be comprised of reusable assets. There are two main reasons to use this approach rather than the big bang approach. Firstly, the up front cost is smaller. As the cost for development of assets can be considered to be divided among the product development projects, the up front cost would only have to cover the pre-analysis and organizational reconstructions. Secondly, the first product will reach the market a lot faster which gives a faster return of investment and builds confidence and commitment for the approach. This type of approach is suitable for a variety of companies as the degree of risk and initial investment can be customized and managed. Thous the prerequisites to chose this approach are much lower than the big bang approach.

If the staff in general does not have much experience with SPL development it would be wise to assign or hire a champion for the initiative, someone who can provide guidance and skills in areas like domain analysis, reference architectures and general SPL concepts. With a champion to lead the initiative, companies that has no previous experience of working with SPL but have extensive knowledge of their domain can learn by doing, get benefits from the SPL, while the risks still remain

manageable. We would recommend this approach for the majority of companies, ranging from companies with no SPL experience but a commitment and willingness to learn to companies with successful product lines who do not wish to put their product release schedule on hold for many man months while dedicating the time solely to asset development. When using this approach type the goal for the SPL practice can be very flexible. Normally the goal would be what Bosch classifies as a Software Product Line maturity or higher, but the goal could be lowered to a Platform or Product Population maturity if the company is not yet ready to take the full step. With this approach the initiative will inevitably linger for a while on either the Platform or the Product Population maturity before enough iterations have been made to complete the majority of the assets.

Incremental with iterative analysis (evolutionary, reactive)

Characteristics: Low risk; Low commitment; Low initial funding; Learning while progressing; Domain does not have to be stable.

An approach that is incremental with iterative analysis is characterized by systematic but opportunistic reuse. The company has taken the business decision to advocate reuse and develop a reusable asset library. There is no overall plan for a SPL and if there is a reference architecture it is usually not enforced and can be modified as necessarily. The development is more oriented towards single system development but commonality and variability analysis in a domain context is carried out in each project to look for subsystems or components that could be made reusable. If this approach is used without having a PLA then you create subsystems with no certain structure. When you are to derive the products you create unofficial product specific architectures for every architecture. This approach could be useful when there is many products that are different which makes the creation of a common PLA which fulfills the quality requirements hard. Consequences of this approach is that maintenance becomes significantly tougher and it also makes automation of product derivation impossible. This approach is referred to by Bosch as Product Population and could be a middle step after having a platform and before having a SPL. But creating the architecture after the assets will probably lead to having to redefine assets to fit with the newly created architecture.

An approach to SPL adoption that is evolutionary and reactive does not really aim for a complete SPL practice and looses many of the potential benefits with SPL. But could be advisable when a company wishes to increase reuse but do not have the confidence to aim for a SPL practice right from the start. There can be several reasons for choosing this type of approach. One reason can be that the commitment for an SPL approach is low or that the staff lacks or is inexperienced in process discipline. Another reason can be that the domain is rapidly changing. In these circumstances a full SPL approach is not advisable. However, introducing reuse and SPL concepts can with time train the staff in process discipline and build understanding and commitment for a SPL approach as well as associated skills. For a small cost at a low risk, this can give the company an advantage if a more complete SPL approach is aimed for in the future. Iterative domain analysis in a rapidly changing domain can be a method for spotting areas of the domain that are becoming stable and can be made reusable. This may give the company a competitive edge. The downside with this type of approach is that the assets easily corrode and that a multitude of variations occurs, making configuration management and version control unmanageable. Each product will likely have to be maintained as a unique system and not a configuration of assets as is the

goal of a SPL approach. As a result the development costs of each product may well decrease with this approach compared to single system development without reuse, the long and costly maintenance phase will not be affected. Choosing this approach type will limit the maturity of the SPL to what Bosch classifies as Platform or Product Population. Reaching and maintaining a higher SPL maturity requires a somewhat stable domain, planning based on domain analysis and process discipline not to be tempted to cut corners.

When choosing an incremental with iterative analysis type of approach there is no adoption approach to follow. As this type of approach is characterized by ad-hoc utilization of SPL techniques and reuse it will become very specific to each organization. A recommendation from our side is to start each new product development project with a feature analysis of the product and compare it to previously developed products. This should be done in order to identify parts of the product that may already exist in earlier products and may be extracted and reused. If the feature analysis is also compared to future products it may also be possible to identity parts of the product that may be reused in the future. If this is the case than the designers and developers of these parts could have this in mind and save themselves redundant work in the future. If there is something that is often reused it would be a good idea to make it easy to reuse. With time it is possible to create a library of ready to reuse components, much like a SPL asset library. A reference architecture is not something that is a necessity with this approach, however it does help when making components and subsystems reusable.

2.2.4.3 Choice of Specific Adoption Approach

In this section we present a small selection of adoption approaches to chose from. We aimed for less then ten approaches to make the choice easier and the differences are discussed below in this section and in Section 2.2.3. These approaches were selected from a much larger number of similar approaches based on the level of detail in the provided descriptions. Several of the adoption approaches can be used as more than one adoption type as described below. The incremental with iterative analysis approach type does not have any adoption approaches. The reason for this and how this approach type is used is discussed in Section 2.2.3.

Incremental with Pre-analysis
- Evolutionary Initiation Approach [SIM02]
- Bosch Group Case Study Approach [THI01]
- Audio & Video Case Study Approach [KIM07a]
- Observation Case Study Approach [LAG04]
- UP Enhanced for Product Line [ZHA06]

Big Bang
- Adoption Factory Pattern [CLE06]
- Quality-driven Architecture Design and Analysis [MAT02]
- Bosch Group Case Study Approach [THI01]
- Audio & Video Case Study Approach [KIM07a]
- Evolutionary Initiation Approach [SIM02]
- Observation Case Study Approach [LAG04]
- UP Enhanced for Product Line [ZHA06]

Incremental with Iterative Analysis
- Ad-hoc. No proposed method to follow (see Section 2.2.3 above).

Adoption Factory Pattern (AFP) [CLE06] guides the adoption process with help of the practice areas of the SEI Framework by stating when in the adoption progress you should use what practice areas and in what practice area certain activities are described. This roadmap should be suitable for organization which chooses the Big Bang approach and thus wants to create a completely new process. This process will then be based on the SEI Framework and its practice areas giving the organization all it should need for it to execute a successful SPL adoption. The AFP could probably be used for incremental transition of your old process as well but you will still need to incorporate the SEI Framework practice areas to fully follow and use this roadmap.

The evolutionary initiation approach proposed by Simon et al. in [SIM02] is a quite standard adoption approach and so is the approach used in the Bosch Group case study [THI01]. Excluding the last phase of the proposed approach [SIM02] makes them very similar and both describes the according to what we found the most common way of introducing SPL. These two is the two we find the best, based on descriptions, plausibility, validation and most used. Both describes the planning phase of the adoption i.e. scoping, analysis, modeling and architecture creation, which makes them suitable for both Incremental with Pre-analysis as well as the Big Bang approach. The last phase in the proposed approach is a reengineering phase which could be included if you want to reconstruct your current architecture and components into a PLA and core assets.

In a case study in the audio and video domain [KIM07a] a similar approach to those two mentioned above was used. It is the most extensively described approach except for in the area of domain, scope and market analysis. We recommend this adoption approach if the organization has a clear understanding of the market and of the domain. If not the organization can add domain, scope and market analysis activities of their own choice to this approach. A small domain and scope definition activity may be needed even if the organization have a strong knowledge of the domain although it could be made more ad hoc. This approach incorporates implementation and testing but should be suitable for either Incremental with Pre-analysis or Big Bang because the difference between these approaches types are in how and when the assets and products are created rather then in the preparations with analysis and design.

UPEPL [ZHA06] is based on UP with incorporated SPL practices and should be the preferred choice for organizations using the UP model. UPEPL can of course be used by other organizations but with less benefits.

Observations from a case study in [LAG04] shows an approach that focuses on reengineering of legacy products. This approach can be chosen if an organization are to develop an SPL with no new features and with legacy products that are all suitable for extracting assets from since it lacks in creation of assets from scratch.

Last of the suggested approaches, Quality-driven Architecture Design and Analysis [MAT02] is a very extensive approach that is architecture heavy and are specifically designed for a revolutionary approach where an architecture and a complete set of assets are created before developing the first product. This approach we recommend if a Big Bang approach is chosen and a thorough architecture with multiple levels and viewpoints is to be created as the first step of the transition.

2.2.4.4 Choice of Framework

Following a framework for the SPL development provides structure and guidelines on how to work. A framework should contain concepts, principles and practices that works as a basis and supports correct SPL development. There exist several frameworks to choose from, some are suggested by researchers others developed internally in organizations. Discussion of the research area of SPL frameworks was covered in 2.2.1.1.

The most famous and most accepted framework is according to our findings the Framework for Software Product Line Production by SEI [NOR07] . This is also the most extensive and most validated according to our findings. There are some other frameworks that are very similar which might work just well although they are not as validated. Two of them are the Generic Product Line Process Framework [VEH00] and the Product Line Engineering Practices Mode [COA05]. These are both based on the SEI Framework but are not as extensive nor as validated which further speaks for the SEI Framework. We recommend the SEI Framework which should be suitable for almost any situation, but any of the three should suffice. Another famous framework is the BAPO or later the FEF, this framework is suitable for evaluating and improving a existing product line. It evaluates the four BAPO dimensions of the organization separately and thus it gives hints on where to improve.

2.2.4.5 Choice of Methods

When it comes to choosing methods for SPL development there is a general rule to remember, do not make it to complicated. As the validation of methods are scarce, validated recommendations can hardly be made (See Section 2.2.1.2). Using common sense and keeping SPL concepts in mind should go a long way in an initial transition. The learning curve for SPL development has been reported to be steep [NOR07] so we recommend an organization who already have development methods that works well for single system development to kept and adapted what they have if possible.

If practices are missing and has to be introduced then try to chose one that fits into and compliment what already exist. That said, it is likely that any introduced method will have to be customized to the specific organization. Keep in mind that the methods are tools, not the goal.

When unsure of what to do or in what order to do it, we recommend using the SEI framework [NOR07] or the PuLSE methodology [Appendix D] as a guide. They are the most complete, well described and validated works on the subject. FEF [LIN05] and related frameworks [LIN02, LIN04] can be used to set goals for the SPL development as well although it is not as detailed on how to perform activities as for example the PuLSE methodology.

The one area that distinguish SPL engineering from single system development is domain engineering. If a big bang or pre-analysis adoption approach was selected the development practice should include a strong scoping process and rigorous domain analysis. These two areas are also reported as success factors for SPL adoption [LEE02, KIM07a, KOS06, JEN07]. For scoping we recommend PuLSE-Eco [Appendix D]. PuLSE-Eco is one of the few methods that stand out as being fairly well validated [KUH09, BAY99, DEB98a, KNA00]. It has been applied in companies of various sizes, including small companies [KNA00], and should be applicable to almost any organization.

For domain analysis we recommend an approach that include feature modeling. Feature modeling is a well used and often praised practice [LEE02, KIM07a, KIR06, TIS07]. There are many proposed feature modeling methods but it is hard to recommend any of them as they all claim to work but lack in validation (see Section 2.2.1.2.2). As the major difference between feature modeling methods is usually a matter of notation and syntax any of them should probably work fine. But if the company lacks experience in feature modeling then FODA [KAN90] is a good starting point as the notation is rather simplistic and many methods extend on FODA or use it's notation. That said, FODA in itself will likely not fulfill all domain analysis needs of your company as it is basically just a notation. [LEE02] contains a really good description of feature modeling concepts as well as some practical guidelines that can be used independently of the method used.

Some of the evaluated methods for domain engineering are describing a more complete method that often includes domain analysis, domain modeling and sometimes architecture and design practices. Methods worth mentioning are FORM [KAN98] and ODM [SIM95, SIM96] as well as a combination of PuLSE-Eco, PuLSE-CDA and PuLSE-DSSA [Appendix D]. Also FeatuRSEB [GRI98] and PLUSS [ERI05, ERI06] looks promising. They all have slightly different focus and are somewhat validated, thous one can not be recommended over the others. However, at least one of these methods should benefit the SPL engineering approach taken at any company. Brief descriptions of these methods, with the exception of PuLSE, can be found in Appendix C. PuLSE methods are described in Appendix D.

2.2.4.6 Additional Development Guidelines

The guidelines in this section are based on lessons learned and general principles for SPL development identified in previous works. In this section we collect the tips and recommendations we have found and wanted to repeat that did not fall under any of the section above.

When it comes to acquiring the assets there exists several options, creating, extracting, purchasing or commissioning [NOR07]. Creating the assets from scratch is perhaps the most common approach. Extracting assets from legacy components is also common. The advantage of these two techniques is that the organization has full control over the assets, their design, implementation and future reengineering as requirements changes [NOR07]. Creating the assets from scratch gives an opportunity to design and implement the assets in a more optimized way while extracting assets from legacy systems may reduce the effort needed to provide the core assets. The last statement requires that the legacy system is sufficiently documented and that the design is rather close to what is desired. Both these approaches takes time from the regular development. A different way to acquire assets is to buy them. This can be done by buying components of the shelf, so called COTS. This can be a cheap way of acquiring good and proven assets but you do not have the same possibility of modifying the assets [NOR07]. The last option is to hire an outside development unit to create the asset for you. This could be considered when the internal development unit has their hand full with the ordinary development. There is almost a requirement for an organization to have a good legal department if to commission assets [NOR07]. If not there is greater risk for legal right problems and unless the source code and all related documentation is handed over, a dependency on the commissioned company may develop, making change in the asset base even harder than it usually is. The commissioned unit may on the other hand have more experience in creating generic

core assets and as said before may enable the internal development unit to hold any previously laid release plans[NOR07].

3 DISCUSSION

In the following section we summarize the answers to our research questions for each of which we provided detailed discussions in Chapter 2. In Section 3.2, we discuss the validity threats in this study.

3.1 Answers to Research Questions

RQ1. How should the SPL practices be structured?

Short answer is to structure the practices according to a framework for SPL, this however raises some further questions. These questions are answered by the sub-questions.

RQ1.1. What available frameworks for SPL development are there?

We found a total of 17 frameworks and of these eight were selected for further analysis. Exclusions reasons were either not finished or incomplete coverage.

RQ1.2. What are the differences between the found frameworks?

Most of the frameworks are built out of practices areas to have in a SPL, there are some other types of frameworks though. Some focuses on management, one focuses on traceability, another focuses on needs. Most of the frameworks that are not specifically for SPL lacks in product line activities like asset creation or domain modelling.

RQ1.3. Which of these can be recommended?

We recommend three that all have complete coverage of the needed SPL practices and groups the activities in appropriate groups. These three frameworks are: the Framework for Software Product Line Production [NOR07], the Generic Product Line Process Framework [VEH00] and the Product Line Engineering Practices Model [COA05]. They are all three very similar but the FSPLP are more thorough.

Final answer to research question: The SPL practices should be structured in groups of activities for certain areas according to one of the three recommended frameworks above.

RQ2. What methods should an organization consider for their SPL development?

To be able to answer this question one must know what areas that are necessary, what methods that exists for these areas and finally which of these that are potentially useful.

RQ2.1. Which are the specific areas of software product lines?

The areas are to start with those areas of regular development such as requirement engineering, quality assurance, testing and development. SPL specific areas are domain analysis and architecture design and creation.

RQ2.2. What available methods are there for each area?

We found a total of 93 methods, 49 in the area of domain analysis, 30 is concerning architecture design and creation and the last 14 are equally divided in the three areas of requirement engineering, development and quality assurance and testing. Those that were evaluated as potentially useful are described in appendixes C and D.

RQ2.3. Which of these can be recommended?

There are few methods that can be recommended due to poorly described way of work and/or lack of validation. Those few recommended are stated in section 2.2.4. For discussion on the methods refer to section 2.2.1.

Final answer to research question:

Those methods to consider for an organization are those in [Appendix C and D].

RQ3. How does an organization adopt a SPL approach?

This is a very complicated question, we start to try to answer this in the thesis. To simplify we break it down into five sub-questions.

RQ3.1. What available adoption approaches are there?

There are a total of 43 adoption approaches both those proposed by researchers and those used in industry. For a thorough discussion on the adoption approaches, refer to Section 2.2.3.3.2. There also exists several adoption types categorizations, these are described in Section 2.2.3.1.

RQ3.2. Are there any differences between the adoption approaches proposed by researchers and those used in industry?

There is practically only one big difference and it is not on the structure but on how well it can be reused. The proposed solutions lack in validation and the approaches used in industry lack in description to such extent that it prevents them from being repeatable.

RQ3.3. What are the differences between specific adoption approaches?

Overall there are some that creates a SPL without an architecture and some that have architecture focus, some perform the transition in small incremental steps while some describes one big expensive step. Those we the best are very similar to each other though, these can be seen as variation of the same approach.

RQ3.4. What variables affect the choice of adoption approach?

Variables to consider to be able to choose appropriate adoption approach are; current level of commitment of staff and management, how much funding that is given to the initiative, the staffs level of knowledge and experience, how much risks that the organization are willing to take and how stable the domain is. In the end it all boils down to how much funding is available and how much of it the organization is willing to risk. These variables are discussed in detail in Section 2.2.4.2.1.

RQ3.5. What can be learned from industry case studies?

In this area there are quite much information, a big fraction of the lessons learned state requirements in form of what has to exist when using a SPL. Refer to [Appendix H] for the lessons learned with descriptions and source case study.

RQ3.6. Which of the found adoption approaches can be recommended?

19 were marked as potentially useful and were summarized in [Appendix F], eight if these were suggested by researchers and eleven were found in industry case studies. Out of these, four from researchers and three from industry were recommended by us in our guidelines on the basis of being understandable, repeatable and complete.

Final answer to research question:

This question is the primary question for the guidelines and are divided into several steps to perform. These are thoroughly explained in Section 2.2.4.

RQ4. What are the available methods on analyzing an organization's readiness towards adopting a SPL approach?

It is little research conducted on this area, we found only five methods for readiness control. These fulfills different purposes, from fast and brief to time

consuming and thorough. The found readiness methods with summaries are presented in [Appendix E], which, when and how to use them are described as step one of our guidelines in Section 2.2.4.

RQ5. What are the consequences of combining SPL development and CMMI process improvement?

We wanted to find all possible consequences and had to look at several different perspectives.

RQ5.1: Are there any similar procedures or concepts between the two initiatives?

Both initiatives have a strong process focus. The SPL initiative focuses on creating and using processes for development. CMMI focuses on defining, evaluating and improving processes. As both are used in software development environments there are naturally some similar areas that are considered. At a quick glance these areas match and overlap very well, but this impression is largely superficial.

RQ5.2. Does the CMMI maturity levels affect the SPL adoption process?

Yes, CMMI maturity level three should be a reasonable minimum requirement for initiating SPL since then there is established guidelines on how to create you processes, a process asset library and the processes are generic. For a more detailed discussion on the subject, refer to Section 2.2.2.2.2.

RQ5.3. Are there any CMMI process areas that have a greater benefit for SPL development than for single system development?

We found that many are more important for SPL development, these can be divided into more important and crucial. Amongst those that are crucial we find Configuration Management, Product Integration and Validation. Those that are not as important but still needs more attention are Requirement Management, Integrated Project Management and Organizational Training. In addition Project Planning and Supplier Agreement Management are important in certain cases.

RQ5.4. Are there any CMMI process areas that are more difficult to apply in SPL or has a negative effect on SPL development?

Since SPL probably is more complex and involves multiple products there are some areas that are harder to apply. These are Configuration Management, Requirements Management, Integrated Project Management, Requirements Development, Validation and Verification. For a complete list of the process areas and their importance in SPL refer to Table 6 and for a discussion on them refer to Appendix G.

RQ5.5. Can the introduction or improvement of SPL practices improve the organization's process maturity as specified by CMMI?

This is not likely, we have found no evidence of such a relationship and no indication that improvement of SPL practices alone will increase the CMMI maturity. None of the related work articles draw any conclusions in this direction either.

Final answer to research question:

There are some studies that indicates positive results from combining these two initiatives. These are on a high abstraction level though. When looking closer on a possible combination the results are less positive and largely inconclusive. We identify some PA's that are more important when considering SPL development and a few others that may be harder to execute. We conclude that SPL benefits form process maturity and discipline as SPL development is process controlled and a lack in process

discipline may cause corrosion of the SPL. We could not identify any drawbacks of having both an CMMI initiative and SPL transition initiative.

RQ6. Can SPL practices be categorized into maturity levels?

SPL can be categorized into several maturity levels, ranging from a platform to a configurable product base. With Product Population, Software Product Line and Programme of Product Line as some maturity levels between them. You do not have to pass them all and there can exist different maturities at different levels of scope. We argue however that these are not "true" maturity levels but rather a set of states that the SPL development could enter. When looking at others which have investigated upon SPL maturity we found that all articles derives from the work by Jan Bosch and his definition of SPL maturity. There are one case study though, that builds on the maturity tree defined by Jan Bosch. They identify several states not described in the original maturity tree. Their version of the tree is however very complex and impossible to use as a maturity tree or ladder. Further discussion on SPL maturity are presented in 2.2.2.

3.2 Validity Threats

There are a few validity threats associated with systematic literature reviews. One of them is that it is hard to tell if all relevant articles were captured. Factors that affect the level of this threat include the sources that are used and constrains put on the search. Using the wrong sources and badly constructed search strings may on the other hand pose serious threats to the validity of a systematic literature review. Constrains include very specific search strings and, like in our case, a time period constrain. Our approach to minimizing this validity threat was twofold. The first part consisted of a thorough prestudy where data sources were examined and the research field was sampled to identify keywords to be used in the search strings. During this prestudy we also made sure that our time period constrain could be considered valid. The second part of our approach to minimizing the validity threat of the methodology was to go through the reference list of the articles gained through the initial part of the literature review. By doing this we were able to catch a few articles that was not included in by the search strings or the time period constrain.

A common critique against the recursive abstraction technique is that the final results will be many layers abstracted from the original data. Thus there is a risk that vital data has been lost in the abstraction. In our case we do not think this is a large issue as we did not do that many recursive iterations. Much of the discussions and conclusions made in this article is based on the second and third recursive abstraction levels. To further minimize this validity threat we kept track of what each summary was based on down to the original articles. This was very helpful as we often went back to the original articles when we needed clarification on an issue.

There are at this point some validity concerns about the guidelines section of this thesis as we have yet to verify and validate them. They are based on the best literature available and lessons learned in case studies, but the validation of the research used is rather low as we will conclude in the next chapter. We identified and contacted a few companies, who's experience and feedback might have been of interest, to ask if they could participate by answering a few questions in the form of a interview or questionnaire. Most of the companies was positive to the study but none was willing to participate. The validation of our guidelines has been added to future works.

4 CONCLUSION

Although Software Product Lines are considered a fairly new phenomena there is a lot of research available on the topic. The majority of the available research are on methods for domain engineering or architecture related research. Most methods claim originality but it is often easy to find a couple of methods addressing the same issue in a similar way. A couple of frameworks has been suggested that address the entire SPL process with a higher level of abstraction then the proposed methods. Most are inspired by or an adoption of the SEI framework. An exception is BAPO which is a framework for evaluating an organization that are using SPL rather than a framework for how to work with SPL.

Unfortunately the entire SPL research field suffers from lack of validation. Most methods claim that the method has been applied in a industrial or scientific project but details of the project and evidence of the methods usability are missing. Industrial case studies and experience reports do sometimes provide these result data needed to validate a method, but then they rarely specifies if they use a specific method. And if the company in the case study or experience report use their own in house developed methods, the methods are usually not described in enough detail to make them repeatable. There may be several reasons why this is so. Companies and organizations who have made investments in their SPL may not want to give to much away and risk loosing any competitive advantages they may have gained. Another possibility is that experiences with methods that have been tested and not given the hoped for results have not been published.

As we investigated how SPL practices were classified we found quite a few papers on SPL maturity. Jan Bosch is the big name in this field as he has authored or co-authored basically all papers of some quality. Although his maturity levels for SPL are generally accepted by the researchers and practitioners we do not believe that they are maturity levels in the same sense as for example CMMI maturity levels. The term maturity is according to us a positive word, indicating that a higher maturity is always to strive for. Unfortunately this is not always the case with SPL maturity defined as by Bosch. A higher maturity does not necessarily mean a better practice. The nature of the domain, organizational structures and the number of product lines used within the company greatly affects what maturity level is optimal. We believe these maturity levels are more like states of development practices that the development can move through as the practices evolve than a measurement of maturity.

CMMI and SPL engineering have at least superficial similarities. This is perhaps not a coincidence since a lot of work on product lines are influenced by the framework for SPL developed by SEI, which also is the institution behind CMMI. The relationship between the two have been investigated before and the result is largely inconclusive. We believe this may be because a direct relationship would be highly desirable but is really hard to find, if it exists at all. To find such a relationship would require an in depth study and possibly result in nothing. Indirect relationships are easier to find but are not as desirable. SPL development requires that the processes are defined and followed since previous investments and effort is at stake. CMMI provides evaluation of this ability. A CMMI initiative at the third maturity or higher should help prevent deterioration of the SPL as specified processes will more likely be followed.

We have found from case studies and experience reports that there is a fairly standard approach to adopt SPL practices. Almost every case describes some variant of what we classified as an incremental with pre-analysis approach. The exception to this is almost exclusively military funded projects. Many cases did start with SPL in a pilot project or in small scale. When the understanding and acceptance for the approach increased the initiative was scaled up. The reason for this may be that there is still a lot of skepticism to product lines among software developers and managers. This may in turn be because of the lacking validity of suggested methods, but it is more likely due to resistance to change in general. According to the experience reports this has been a successful way of introducing SPL. The initial goals for the initiative should not be set to high before this first trial period has passed. To create acceptance and commitment may be a goal in itself.

In this thesis we have made the to our knowledge first systematic literature review of the Software Product Line research field as a whole. We have identified some areas that have been fairly well researched and areas where there is a lack in research or in validation of the existing research. Our work could be used as an entry point to the research field or as a stepping stone for further research.

By using and combining previous research we have constructed a set of guidelines that we hope will make a transition to SPL development easier and make the learning curve a bit less steep. There are previous works that are partially similar. But they either have a much broader scope like a framework for SPL development and are thous more abstract or they have a much smaller focus like PLPA and the Product Line Technical Probe. In our work we make use of the later mentioned works and use the former mentioned works as inspiration for what to achieve in the end.

4.1.1 Future work

We suggest the work on a benchmarking tool for comparison of SPL methods. This would a common way of evaluating the methods and thus giving those found usable some very needed validation. These methods should then be subject of case studies to increase the validation. The case studies should specify what method is used and preferably provide quantitative data for the results. Without this it is extremely hard to provide more specific guidelines and recommendations.

To validate and improve our guidelines we want to perform a thorough case study on our work. This case study should be performed on several organizations with different situations who introducing a SPL approach with help of our guidelines. It is important that they have different starting points to test more than one of our recommended approaches. This would enlighten eventual flaws and gives us a chance to improve our guidelines.

The research on the combination of SPL and CMMI are as previously stated mostly on a high level and we see the need for a deeper analysis on the subject. This would clear the confusion on the area and provide a definite answer to the question if there are any gain or benefits to be had by combining the two initiatives. The same could be done with SPL and ISO-TickIt for the same reason.

5 ACKNOWLEDGMENTS

We would like to start by thanking our supervisor Cigdem Gencel, because without her guidance, the thesis would not be what it is. We also want to thank Blekinge Institute of Technology for the opportunity to work with such a interesting area.

6 REFERENCES

[ABI07] Marwan Abi-Antoun, "Making frameworks work: a project retrospective", *Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*, Pages: 1004 – 1018, ACM, 2007

[AHM06] Faheem Ahmed and Luiz Fernando Capretz, ”Maturity Assessment Framework for Business Dimension of Software Product Family”, *International Journal of Interpretability in Business Information Systems*, Volume: 1, Number: 1, Pages: 9-32, 2006

[AHM06] Faheem Ahmed and Luiz Fernando Capretz, “A decision support tool for assessing the maturity of the software product lines”, *International Journal of Computing Information Sciences*, Volume: 4, Number: 3, Pages, The APCEP, 2006.

[AHM08a] Faheem Ahmed , Luiz Fernando Capretz and Jagath Samarabandu, ”Fuzzy inference system for software product family process evaluation”, *Information Sciences: an International Journal*, Volume: 178, Issue: 13, Pages: 2780-2793, Elsevier Science Inc, 2008

[AHM08b] Faheem Ahmed, Luiz Fernando Capretz and Muhammad Ali Babar, "A Model of Open Source Software-Based Product Line Development", *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, Pages: 1215-1220, IEEE Computer Society, 2008.

[AHM10] Faheem Ahmed and Luiz Fernando Capretz, "An organizational maturity model of software product line engineering", *Software Quality Control*, Pages: 195-225, Kluwer Academic Publishers, 2010.

[ALF08] Mauricio Alférez, Uirá Kulesza, André Sousa, João Santos, Ana Moreira, João Araújo and Vasco Amaral, ”A model-driven approach for software product lines requirements engineering”, *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, Pages: 779–784, Knowledge Systems Institute Graduate School, 2008.

[ALI07] Fabrice Alizon, Kiran Khadke, Henri J. Thevenot, John K. Gershenson, Tucker J. Marion, Steven B. Shooter and Timothy W. Simpson, ”Frameworks for Product Family Design and Development”, *Concurrent Engineering*, Volume: 15, Number: 2, Pages:187-199, 2007

[ALI09] Muhammad Sarmad Ali, Muhammad Ali Babar and Klaus Schmid, "A Comparative Survey of Economic Models for Software Product Lines", *Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications*, Pages: 275-278, 2009.

[ALV06] Vander Alves, Rohit Gheyi, Tiago Massoni, Uirá Kulesza, Paulo Borba and Carlos Lucena”Refactoring product lines”, *Proceedings of the 5th international conference on Generative programming and component engineering*, Pages: 201 – 210, ACM, 2006

[AME01] Pierre America, Steffen Thiel, Stefan Ferber and Martin Mergel, ”Introduction to Domain Analysis”, Eureka S! 2023 Programme, ITEA project 99005, 2001, http://www.esi.es/esaps/public-pdf/CWD121-20-06-01.pdf, accessed 2010-07-18.

[ANA00] Michalis Anastasopoulos, Joachim Bayer, Oliver Flege, Cristina Gacek, ”A Process for Product Line Architecture Creation and Evaluation. PuLSE-DSSA - Version 2.0”, IESE-Report, 038.00/E, Fraunhofer IESE, 2000.

[ANQ09] Nicolas Anquetil, Uirá Kulesza, Ralf Mitschke, Ana Moreira, Jean-Claude Royer, Andreas Rummler and André Sousa, "A model-driven traceability framework for software product lines", *4th Traceability Workshop held in conjunction with European Conference on Model-Driven Architecture*, Springer, 2009.

[AOY08] Mikio Aoyama and Atsuko Yoshino, ”AORE (aspect-oriented requirements engineering) methodology for automotive software product lines”, *Proceedings of the 2008 15th Asia-Pacific Software Engineering Conference*, Pages: 203-210, IEEE Computer Society, 2008

[ARD00a] Mark Ardis, Nigel Daley, Daniel Hoffman, Harvey Siy and David Weiss, ”Software product lines:a case study ”, *Software—Practice & Experience*, Volume 30, Issue 7, Pages: 825 – 847, John Wiley & Sons Inc, 2000

[ARD00b] Mark Ardis, Peter Dudak, Liz Dor, Wen-jenq Leu, Lloyd Nakatani, Bob Olsen and Paul Pontrelli, "Domain engineered configuration control", *Proceedings of the first conference on Software product lines : experience and research directions: experience and research directions,* Pages: 479-493, Kluwer Academic Publishers, 2000.

[ASA09] Mohsen Asadi, Bardia Mohabbati, Nima Kaviani, Dragan Gašević, Marko Bošković and Marek Hatala, "Model-driven development of families of Service-Oriented Architectures ", *Proceedings of the First International Workshop on Feature-Oriented Software Development*, Pages: 95-102, ACM, 2009

[ASI06] Timo Asikainen, Tomi Mannisto and Timo Soininen, "A unified conceptual foundation for feature modelling", *Proceedings of the 10th International on Software Product Line Conference*, Pages: 31-40, IEEE Computer Society, 2006.

[ATK00] Colin Atkinson, Joachim Bayer and Dirk Muthig, "Component-Based Product Line Development The KobrA Approach", Proceedings of the first conference on Software product lines : experience and research directions: experience and research directions, Pages: 289-309, Kluwer Academic Publishers, 2000.

[BAB07] Muhammad Ali Babar, "Evaluating Product Line Architectures Methods and Techniques", Proceedings of the 14th Asia-Pacific Software Engineering Conference, Page: 13, IEEE Computer Society, 2007.

[BAN00] Sergio Bandinelli and Goiuri Sagardui Mendieta, “Domain Potential Analysis: Calling the Attention on Business Issues of Product Lines”, *Proceedings of the International Workshop IW-SAPF-3*, Pages: 76-81, Springer, 2000.

[BAS05] Rabih Bashroush, John Brown, Ivor Spence and Peter Kilpatrick, "ADLARS An architecture description language for software product lines", *Proceeding of the 29th Annual IEEE/NASA Software Engineering Workshop of 2005*, Pages: 163-173, IEEE Computer Society, 2005.

[BAY99a] Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen and Jean-Marc DeBaud, "PuLSE: A Methodology to Develop Software Product Lines", P*roceedings of the 1999 symposium on Software reusability*, Pages: 122-131, ACM, 1999.

[BAY99b] Joachim Bayer, Dirk Muthig and Tanya Widen, ”Customizable Domain Analysis”, *Proceedings of the First International Symposium on Generative and Component-Based Software Engineering*, Pages: 178-194, Springer, 1999.

[BAY00a] Joachim Bayer, Oliver Flege, Cristina Gacek , ”Creating Product Line Architectures”, *Proceedings of the International Workshop on Software Architectures for Product Families*, Pages: 210-216, Springer, 2000.

[BAY00b] Joachim Bayer , Cristina Gacek , Dirk Muthig , Tanya Widen, ”PuLSE-I: Deriving Instances from a Product Line Infrastructure”, *Proceedings of the Seventh IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2000)*, Pages:237-245, 2000.

[BAY01] Joachim Bayer, Dirk Muthig and Brigitte Göpfert, ”The library system product line. A KobrA case study”, Kaiserslautern, Technical Report IESE-Report No. 024.01/E, Fraunhofer Institute for ExperimentalSoftware Engineering, Kaiserslautern, Technical Report IESE-Report No. 024.01/E, 2001

[BAY06] Joachim Bayer, Mathias Kose and Alexis Ocampo, "Improving the Development of e-Business Systems by Introducing Process-Based Software Product Lines”, *Product-Focused Software Process Improvement*, Volume 4034/2006, Pages: 348-361, Springer, 2006

[BIG09] BigLever's Inc., “BigLever's SPL Lifecycle Framework”, http://www.biglever.com/solution/framework.html, Last visit on 2010-07-07.

[BIR02] Andreas Birk, ”Three Case Studies on Initiating Product Lines: Enablers and Obstacles”, *Paper presented at the PLEES'02*, 2002

[BIR03] Andreas Birk, Gerald Heller, Isabel John, Klaus Schmid, Thomas von der Maßen and Klaus Müller, ”Product Line Engineering: The State of the Practice”, *IEEE Software*, Volume: 20, Number: 6, Pages: 52-60, 2003

[BOS00] Jan Bosch, "Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach", ACM Press/Addison-Wesley Publishing Co., 2000.

[BOS01] Jan Bosch “Adopting Software Product Lines: Approaches, Artefacts and Organization”, *Proceedings of the International Workshop on Product Line Engineering - The Early Steps: Planning, Modeling, and Managing*, Pages: 19-23, Fraunhofer IESE, 2001.

[BOS02] Jan Bosch, “Maturity and Evolution in Software Product Lines; Approaches, Artifacts and Organization”, P*roceedings of the Second Conference Software Product Line Conference*, Page 257-271, Springer, 2002.

[BOS03] Jan Bosch, ”Maturing Architectures and Components in Software Product Lines”, *Component-Based Software Quality*, Pages: 246-258, Springer, 2003

[BOS04] Jan Bosch, ”On the Development of Software Product-Family Components”, *Software Product Lines*, Volume 3154/2004, Pages169-173, Springer Berlin, 2004

[BOS05] Jan Bosch, “Staged adoption of software product families“, *Software Process Improvement and Practice*, Volume 10, Issue 2, Pages 125-142, John Wiley & Sons, Ltd. 2005.

[BOS07] Jan Bosch, ”Invited talk: expanding software product families: from integration to composition ”, *Proceedings of the 20th international conference on Architecture of computing systems*, Pages: 283-295, Springer, 2007

[BOT09] Goetz Botterweck, Andreas Pleuss, Andreas Polzer and Stefan Kowalewski, ”Towards feature-driven planning of product-line evolution”, *Proceedings of the First International Workshop on Feature-Oriented Software Development*, Pages: 109-116, ACM, 2009

[BRE05] Michael Breen, "Experience of using a lightweight formal specification method for a commercial embedded system product line", *Requirements Engineering*, Volume 10, Issue 2, Pages: 161-172, Springer, 2005.

[BÜH04] Stan Bühne, Gary Chastek, Timo Käkölä, Peter Knauber, Linda Northrop and Steffen Thiel, “Exploring the Context of Product Line Adoption”, *Software Product-Family Engineering*, Page 19-31, Springer Berlin / Heidelberg, 2004.

[BUH05] Stan Buhne, Kim Lauenroth and Klaus Pohl, "Modelling Requirements Variability across Product Lines ", *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Pages: 41 – 52, IEEE Computer Society, 2005

[BÖC02] Günter Böckle, Jesús Bermejo Muñoz, Peter Knauber, Charles W. Krueger, Julio Cesar Sampaio do Prado Leite, Frank van der Linden, Linda Northrop, Michael Stark and David M. Weiss, “Adopting and Institutionalizing a Product Line Culture”, *Proceedings of the Second Conference Software Product Line Conference*, Page 49-59, Springer, 2002.

[CAP06] Mauro Caporuscio, Henry Muccini, Patrizio Pelliccione and Ezio Di Nisio, ”Rapid System Development Via Product Line Architecture Implementation”, *Rapid Integration of Software Engineering Techniques*, VolumeVolume 3943/2006, Pages18-33, Springer Berlin, 2006

[CAT08] Cagatay Catal and Banu Diri, "A Conceptual Framework to Integrate Fault Prediction Sub-Process for Software Product Lines", *Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, Pages: 99-106, IEEE Computer Society, 2008.

[CER04] Rodrigo Cerón, José L. Arciniegas, José L. Ruiz, Juan C. Dueñas, Jesús Bermejo and Rafael Capilla, "Architectural modelling in product family context", *Proceedings of the First European Workshop on Software Architecture*, Volume: 3047, Pages: 25-42, Springer, 2004.

[CHA04] Gary Chastek, Patrick Donohoe and John D. McGregor, “A Study of Product Production in Software Product lines”, Software Engineering Institute, 2004.

[CHI06] Adam Childs, Jesse Greenwald, Georg Jung, Matthew Hoosier and John Hatcliff, "CALM and Cadena metamodeling for component-based product-line development", *Computer*, Volume: 39, Number: 2, Pages: 42-50, IEEE Computer Society, 2006.

[CLE01] Paul Clements and Linda Northrop, ”Software Product Lines: Practices and Patterns”, Addisson-Wesley, 2001

[CLE02] Paul Clements, Rick Kazman and Mark Klein, “Evaluating Software Architectures”, Addison-Wesley Professional, 2001.

[CLE05a] Paul C. Clements, John McGregor and Sholom G. Cohen, ”The Structured Intuitive Model for Product Line Economics (SIMPLE)”, *TECHNICAL REPORT CMU/SEI-2005-TR-003 ESC-TR-2005-003,* February 2005

[CLE05b] Paul Clements, "A competition of software product line economic models", *Proceeding of the International Conference of Software Product Lines,* Page: 136, Springer, 2005.

[CLE06] Paul Clements, Lawrence Jones, John McGregor and Linda Northrop, “Getting there from here - A Roadmap for software product line adoption”, *Communications of the ACM*, Volume 49, Issue 12, Page 33-36, ACM, 2006.

[COA05] François Coallier and Roger Champagne, “A product line engineering practices model”, *Science of Computer Programming 57*, Page: 73–87, 2005.

[DEB98a] Jean-Marc DeBaud and Peter Knauber, **”**Applying PuLSE for Software Product Line Development”, *Proceedings of the European Reuse Workshop '98*, Madrid, November 1998.

[DEB98b] Jean-Marc DeBaud, Oliver Flege, Peter Knauber, ”PuLSE-DSSA—a method for the development of software reference architectures”, *Proceedings of the third international workshop on Software architecture*, Pages: 25 – 28, ACM, 1998.

[DEB99] Jean-Marc DeBaud and Klaus Schmid, ”A systematic approach to derive the scope of software product lines”, *Proceedings of the 21st international conference on Software engineering*, Pages: 34 – 43, ACM, 1999.

[DEE04] Sybren Deelstra, Marco Sinnema, Jos Nijhuis and Jan Bosch, "COSVAM: A Technique for Assessing Software Variability in Software Product", *Proceedings of the 20th IEEE International Conference on Software Maintenance*, Configuration in Industrial Product Families, 2004.

[DEE05] Sybren Deelstra, Marco Sinnema and Jan Bosch, ”A Product Derivation Framework for Software Product Families”, *Journal of Systems and Software*, Volume: 74, Issue: 2, Pages: 173 – 194, Elsevier Science Inc, 2005.

[DEN05] Norman K. Denzin and Yvonna S. Lincoln, "The SAGE Handbook of Qualitative Research", Sage Publications Inc., 2005.

[DHU08] Deepak Dhungana, Thomas Neumayer, Paul Grunbacher and Rick Rabiser, "Supporting Evolution in Model-Based Product Line Engineering", *Proceedings of the 2008 12th International Software Product Line Conference*, Pages: 319-328, IEEE Computer Society, 2008

[DIA00] Jorge L. Diaz-Herrera, Peter Knauber and Giancarlo Succi, ”ISSUES AND MODELS IN SOFTWARE PRODUCT LINES”, *International Journal of Software Engineering and Knowledge Engineering*, Volume: 10 Number: 4 Pages: 527-539, World Scientic Publishing Company, 2000

[DIK01] David M. Dikel, David Kane and James R. Wilson, "Software Architecture: Organizational Principles and Patterns", Prentice-Hall, 2001.

[DJE06] Olfa Djebbi and Camille Salinesi, "Criteria for Comparing Requirements Variability Modeling Notations for Product Lines", Proceedings of the Fourth International Workshop on Comparative Evaluation in Requirements Engineering, Pages: 20-35, IEEE Computer Society, 2006.

[DJE07a] Olfa Djebbi and Camille Salinesi ”RED-PL, a Method for Deriving Product Requirements from a product line requirements model”, *Proceedings of the 19th international conference on Advanced information systems engineering*, Pages: 279-293, Springer, 2007.

[DJE07b] Olfa Djebbi, Camille Salinesi and Gauthier Fanmuy, ”Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues”, *15th IEEE International In Requirements Engineering Conference, Requirements Engineering Conference*, Pages: 301-306, IEEE Computer Society, 2007

[DOB00] Liliana Dobrica and Eila Niemelä, “Attribute-Based Product-Line Architecture Development for Embedded Systems”, *Proceedings of the 3rd Australasian Workshop on Software and Systems Architectures*, Pages: 76-88, IEEE, 2000.

[EBE02] Christof Ebert, "Requirements Management in a Product Line Scenario", *10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02)*, Page: 123, 2002

[EBE03] Christof Ebert and Michel Smouts, ”Tricks and traps of initiating a product line concept in existing products”, *Proceedings of the 25th International Conference on Software Engineering*, Pages: 520 – 525, ACM, 2003

[EGY00] Alexander Egyed, Nikunj Mehta and Nenad Medvidovíc, ”Software Connectors and Refinement in Family Architectures ”, *Software Architectures for Product Families*, Volume 1951/2000, Pages96-106, Springer Berlin, 2000

[ERI05] Magnus Eriksson, Jürgen Börstler and Kjell Borg, ”The PLUSS Approach - Domain Modeling with Features, Use Cases and Use Case Realizations Software product line modeling made practical”, *Software Product Lines*, Volume 3714, Pages: 33-44, Springer, 2005.

[ERI06] Magnus Eriksson, Jürgen Börstler and Kjell Borg, ”Software product line modeling made practical”, *Communications of the ACM*, Volume 49, Issue 12, Pages: 49-54, ACM, 2006.

[FAU01] Stuart R. Faulk ”Product-Line Requirements Specification (PRS): An Approach and Case Study”, *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, Page: 48, IEEE Computer Society, 2001.

[FEL03] Ryan Fellini, Michael Kokkolaras, and Panos Y. Papalambros, "A Rigorous Framework for Making Commonality and Modularity Decisions in Optimal Design of Product Families", *Proceedings of the International Conference on Engineering Design*, 2003.

[FEN05] Qian Feng and Robyn R. Lutz, "Bi-directional safety analysis of product lines", Journal of Systems and Software, Volume 78, Issue 2, Pages: 111-127, Elsevier Science Inc., 2005.

[FER02] Stefan Ferber, Jürgen Haag and Juha Savolainen, ”Feature Interaction and Dependencies: Modeling features for reengineering a legacy product line”, *Proceedings of the Second International Conference on Software Product Lines*, Pages: 235-256, Springer, 2002.

[FEY02] Dániel Fey Róbert Fajta András Boros, ”Feature Modeling: A Meta-Model to Enhance Usability and Usefulness”, *Proceedings of the Second International Conference on Software Product Lines*, Pages: 198-216, Springer, 2002.

[FRI04] Claudia Fritsch and Ralf Hahn, “Product Line Potential Analysis”, *Proceedings of the Third International Software Product Line Conference*, Pages: 228-237, Springer, 2004.

[FRI05] Claudia Fritsch and Burkhardt Renz, ”Four mechanisms for adaptable systems: a meta-level approach to building a software product line ”, *Software Process: Improvement and Practice*, Volume: 10, Issue: 2, Pages: 103-124, 2005

[GAC01] Critina Gacek and Michalis Anastasopoules, ”Implementing product line variabilities”, *ACM SIGSOFT Software Engineering Notes*, Volume 26 , Issue 3, Pages: 109-117, ACM, 2001.

[GAD09] Bruno Gadelha, Ingrid Nunes, Hugo Fuks and Carlos De Lucena, "An approach for developing groupware product lines based on the 3c collaboration model", *Proceedings of the 15th international conference on Groupware: design, implementation, and use*, Pages: 328-343, Springer, 2009.

[GAN07] Dharmalingam Ganesan, Jens Knodel, Ronny Kolb, Uwe Haury and Gerald Meier, ”Comparing costs and benefits of different test strategies for a software product line A study from testo AG , Testing framework-based software product lines” *Proceedings of the 11th International Software Product Line Conference*, Pages: 74-83, IEEE, 2007.

[GOM02] Hassan Gomaa and Michael Eonsuk Shin, "Multiple-View Meta-Modeling of Software Product Lines" *Eighth IEEE International Conference on Engineering of Complex Computer Systems* (ICECCS'02), Page: 238, IEEE Computer Society, 2002

[GOM06] Hassan Gomaa and Mazen Saleh, ”Software Product Line Engineering and Dynamic Customization of a Radio Frequency Management System ”, *Proceedings of the IEEE International Conference on Computer Systems and Applications*, Pages: 345-352, IEEE Computer Society, 2006

[GOM07] Hassan Gomaa, Michael E. Shin, "Automated Software Product Line Engineering and Product Derivation," *40th Annual Hawaii International Conference on System Sciences*, Pages: 285a, IEEE Computer Society, 2007.

[GRE03] Jack Greenfield and Keith Short, ”Software factories: assembling applications with patterns, models, frameworks and tools”, *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Pages: 16 – 2, ACM, 2003

[GRI98] Martin Griss, John Favaro and Massimo d’Alessandro, ”Integrating Feature Modeling with the RSEB”, *Proceedings of the Fifth International Conference on Software Reuse*, Page: 76, IEEE Computer Society, 1998.

[GRO08] Iris Groher, Markus Völter and Christa Schwanninger, "Integrating Models and Aspects into Product Line Engineering", *Proceedings of the 2008 12th International Software Product Line Conference*, Page: 355, IEEE Computer Society, 2008

[GRU08a] Alexander Gruler, Martin Leucker and Kathrin Scheidemann, "Modeling and Model Checking Software Product Lines ", *Proceedings of the 10th IFIP WG 6.1 international conference on Formal Methods for Open Object-Based Distributed Systems*, Pages: 113 – 131, Springer, 2008

[GRU08b] Alexander Gruler, Martin Leucker and Kathrin Scheidemann, "Calculating and Modeling Common Parts of Software Product Lines", Proceedings of the 12th International Software Product Line Conference, Pages: 203-212, IEEE Computer Society, 2008.

[HAB07] Ibrahim Habli and Tim Kelly, "Challenges of establishing a software product line for an aerospace engine monitoring system", Proceedings of the 11th International Software Product Line Conference, Pages: 193-202, IEEE Computer Society, 2007.

[HAL08] Günter Halmans, Klaus Pohl and Ernst Sikora, "Documenting application-specific adaptations in software product line engineering", *Proceedings of the 20th International Conference on Advanced Information Systems Engineering*, Volume: 5074, Pages: 109-123, Springer, 2008.

[HAN08] Geir Hanssen and Tor Fægri "Process fusion: an industrial case study on agile software product line engineering", *The Journal of Systems and Software*, Volume 81, Issue 6, Page: 843-854, Elsevier Science Inc., 2008.

[HAU05] Øystein Haugen, Birger Møller-Pedersen and JonOldevik, "Comparison of system family modeling approaches", *Proceedings of the 9th International Software Product Line Conference,* Volume: 3714, Pages: 102-112, Springer, 2005.

[HET06] William A. Hetrick, Charles W. Krueger and Joseph G. Moore, "Incremental return on incremental investment Engenio's transition to software product line practice", *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, Page: 798-804, ACM, 2006.

[ISH07] Yuzo Ishida, "Software Product Lines Approach in Enterprise System Development", *Proceedings of the 11th International Software Product Line Conference*, Pages: 44-53, IEEE Computer Society , 2007

[JAC97] Ivar Jacobson, Martin Griss and Patrik Jonsson, "Software Reuse: Architecture, Process, and Organization for Business Success", ACM Press/Addison-Wesley Publishing Co., 1997.

[JAI06] Abhinandan Jain and Jeffrey Biesiadecki, "YAM - A framework for rapid software development", *Proceedings of the Second IEEE International Conference on Space Mission Challenges for Information Technology*, Page: 182-194, 2006.

[JAR04] Michel Jaring and Jan Bosch, "Expressing product diversification - Categorizing and classifying variability in software product family engineering", *International Journal of Software Engineering and Knowledge Engineering*, Volume: 14, Issue: 5, Pages: 449-470, World Scientific Publishing Company 2004.

[JAR06] Stan Jarzabek, Bo Yang and Sok Yoeun,"Addressing quality attributes in domain analysis for product lines", *IEE Proceedings on Software of 2006*, Volume: 153, Issue: 2, Pages: 61-73, IET, 2006.

[JEN07] Paul Jensen, "Experiences with product line development of multi-discipline analysis software at Overwatch Textron systems", *Proceedings of the 11th International Software Product Line Conference*, Pages: 35-43, IEEE Computer Society 2007.

[JEP07] Hans Peter Jepsen, Jan Gaardsted Dall and Danilo Beuche, "Minimally Invasive Migration to Software Product Lines", *Proceedings of the 11th International Software Product Line Conference*, Pages: 203-211, IEEE Computer Society, 2007

[JIA08] Michael Jiang, Jing Zhang, Hong Zhao and Yuanyuan Zhou, "Maintaining software product lines — an industrial practice", *IEEE International Conference on Software Maintenance*, Pages**:** 444 – 447, IEEE Computer Society, 2008

[JIR09] Waraporn Jirapanthong and Andrea Zisman, ”XTraQue: traceability for product line systems”, *Software and Systems Modeling,* Volume 8, Number 1, Pages117-144, Springer, 2009

[JOH06] Isabel John, Jens Knodel, Theresa Lehner, Dirk Muthig, "A Practical Guide to Product Line Scoping", *Proceedings of the 10th International Software Product Line Conference,* Pages: 3-12, 2006.

[JON04] Lawrence G. Jones, ”Software Process Improvement and Product Line Practice: Building on Your Process Improvement Infrastructure”, *Technical Note CMU/SEI-2004-TN-04*, 2004

[JON05] Lawrence Jones and Linda Northrop, “Product Line Adoption in a CMMI Environment”, Technical Note: CMU/SEI-2005-TN-028, Carnegie Mellon University, 2005.

[KAN90] Kyo Kang, Sholom Cohen, James Hess, William Novak and Spencer Peterson, ”Feature-Oriented Domain Analysis (FODA) Feasibility Study”, *Technical Report CMU/SEI-90-TR-21*, Software Engineering Institute, Carnegie Mellon University, 1990.

[KAN98] Kyo Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin and Moonhang Huh, ”FORM: A feature-oriented reuse method with domain-specific reference architectures”, *Annals of Software Engineering*, Volume: 5, Number: 1, Pages: 143-168, Springer, 1998.

[KAN05] Kyo Chul Kang, Moonzoo Kim, Jaejoon Lee and Byungkil Kim, ”Feature-Oriented Re-engineering of Legacy Systems into Product Line Assets – a Case Study ”, *Software Product Lines*, Volume 3714/2005 , Pages45-56, Springer, 2005

[KAN07] Sungwon Kang, Jihyun Lee, Myungchul Kim and Woojin Lee, ”Towards a Formal Framework for Product Line Test Development ”, *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, Pages: 921-926, IEEE Computer Society, 2007

[KAU03] Raine Kauppinen, ”Testing framework-based software product lines”, Department of Computer Science, University of Helsinki, 2003

[KHU09] Mahvish Khurum and Tony Gorschek, ”A systematic review of domain analysis solutions for product lines”, *Journal of Systems and Software*, Volume: 82, Issue: 12, Pages: 1982-2003, Elsevier Science Inc, 2009.

[KIM05a] Soo Dong Kim, Soo Ho Chang, and Hyun Jung La, “A systematic process to design product line architecture”, Proceedings from the International Conference on Computational Science and Its Applications, Pages: 46-56, Springer, 2005.

[KIM05b] Soo Dong Kim, Hyun Gi Min, Jin Sun Her and Soo Ho Chang, "DREAM a practical product line engineering using model driven architecture", *Proceedings of the Third International Conference on Information Technology and Applications*, Volume: 1, Pages: 70-75, IEEE Computer Society, 2005.

[KIM06a] Soo Dong Kim, Hyun Gi Min, Jin Sun Her and Soo Ho Chang, “An extreme approach to automating software development with CBD, PLE and MDA integrated”, *Proceedings of PROFES*, Springer, 2006.

[KIM06b] Young-Gab Kim, Jin-Woo Kim, Sung-Ook Shin and Doo-Kwon Baik, ”Managing Variability for Software Product-Line”, *Fourth International Conference on Software Engineering Research, Management and Applications*, IEEE Computer Society, Pages: 74 – 81, 2006

[KIM07a] Kangtae Kim, Hyungrok Kim and Woomok Kim, “Building software product line from the legacy systems experience in the digital audio & video domain”, *Proceeding of the 11th International Software Product Line Conference*, Page: 171-180, IEEE Computer Society, 2007.

[KIM07b] Minseong Kim, Sooyong Park, Vijayan Sugumaran and Hwasil Yang, ”Managing requirements conflicts in software product lines: A goal and scenario based approach”, *Data & Knowledge Engineering*, Volume 61, Issue 3, Pages 417-432, 2007

[KIM08] Taeho Kim, In-young Ko, Sung-won Kang and Dan-hyung Lee ”Extending ATAM to assess product line architecture”, *Proceedings of the 8th IEEE International Conference on Computer and Information Technology*, Pages: 790-797, 2008.

[KIR06] Michael Kircher, Christa Schwanninger and Iris Groher, “Transitioning to a Software Product Family Approach - Challenges and Best Practices”, *Proceedings of the 10th International Conference on Software Product Line*, Page 163-171, IEEE Computer Society, 2006.

[KIS02] Tomoji Kishi, Natsuko Noda and Takuya Katayama, "A Method for Product Line Scoping Based on a Decision-Making Framework", *Proceedings of the 2nd International Conference on Software Product Lines*, Pages: 348-365, Springer, 2002.

[KIT04] Barbara Kitchenham, “Procedures for Performing Systematic Literature Reviews”, Technical Report: TR/SE-0401, ISSN:1353-7776, National ICT Australia Ltd. 2004.

[KNA00] Peter Knauber, Dirk Muthig, Klaus Schmid, Tanya Widen, "Applying Product Line Concepts in Small and Medium-Sized Companies," *IEEE Software,* Volume: 17, Number: 5, Pages: 88-95, 2000.

[KOS06] Eric Koski and Charles Linn, “The JTRS Program: Software-Defined Radios as a Software Product Line”, *Proceedings of the 10th International on Software Product Line Conference*, Page: 182-191, IEEE Computer Society, 2006.

[KRS08] Martin Krsek, Jay van Zyl, Robert Redpath, Ben Clohesy and Nick Dean, "Experiences of large banks Hurdles and enablers to the adoption of software product line practices in large corporate organisations", *Proceedings of the 2008 12th International Software Product Line Conference*, Pages: 161-169, IEEE Computer Society, 2008.

[KRU02a] Charles Krueger, “Easing the transition to software mass customization”, *Revised Papers from the 4th International Workshop on Software Product-Family Engineering*, Pages 282–293,Springer, 2001.

[KRU02b] Charles Krueger, “Eliminating the Adoption Barrier”, *IEEE Software*, Volume: 19, Issue: 4, Pages: 29-31, IEEE Computer Society, 2002.

[KRU06] Charles W. Krueger, ”New methods in software product line development”, *10th International Software Product Line Conference*, Pages: 95 – 99, IEEE Computer Society, 2006

[KRU08a] Charles Krueger, Dale Churchett and Ross Buhrdorf, “HomeAway's transition to software product line practice Engineering and business results in 60 days”, *Proceedings of*

the 12th International Software Product Line Conference, Pages: 297-306, IEEE Computer Society, 2008.

[KRU08b] Charles W. Krueger, ”The BigLever Software Gears Unified Software Product Line Engineering Framework”, *Proceedings of the 2008 12th International Software Product Line Conference*, Page: 353, IEEE Computer Society, 2008

[KÄN05] Kari Känsälä, Piergiorgio Di Giacomo, Jason Mansell, Piero Gutierrez, Guenter Boeckle and Annette Schreiber, “FAMILIES Consortium-Wide Deliverable 2.1: Family Maturity Framework (FMF)”, ITEA/FAMILIES Project, 2005.

[LAG02a] Patricia Lago, “The UNIK project: from product family to product line”, Technical Report PDTDAI-PI-P-001-B0, 2002.

[LAG02b] Patricia Lago and Mari Matinlassi, "The WISE Approach to Architect Wireless Services ", *Proceedings of the 4th International Conference on Product Focused Software Process Improvement,* Pages: 367 – 382, Springer, 2002

[LAG04] Patricia Lago and Hans van Vliet, “Observations from the Recovery of a Software Product Family”, P*roceedings of the International Software Product Line Conference*, Springer, 2004.

[LAG07] Miguel A. Laguna, Bruno González-Baixauli and José M. Marqués, ”Seamless development of software product lines ”, *Proceedings of the 6th international conference on Generative programming and component engineering*, Pages: 85 – 94, ACM, 2007

[LAM05] Sana Ben Abdallah Ben Lamine, Lamia Labed Jilani and Henda Hajjami Ben Ghezala, "A Software Cost Estimation Model for a Product Line Engineering Approach: Supporting tool and UML Modeling", *Proceedings of the Third ACIS International Conference on Software Engineering Research, Management and Applications*, Pages: 383-391, IEEE Computer Society, 2005.

[LAW02] Lawrence G. Jones and Albert Soule, ”Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice”, *Technical Note CMU/SEI-2002-TN-012*, 2002

[LEE02] Kwanwoo Lee, Kyo Chul Kang and Jaejoon Lee, ”Concepts and Guidelines of Feature Modeling for Product Line Software Engineering”, *Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools*, Pages: 62 – 77, Springer, 2002.

[LEE04] Jaejoon Lee, Kyo C. Kang and Sajoong Kim, "A Feature-Based Approach to Product Line Production Planning", *Proceedings of the 3rd International Conference on Software Product Lines*,Volume: 3154, Pages 137-140, Springer, 2004.

[LEE09] Hyesun Lee, Hynsik Choi, Kyo C. Kang, Dohyung Kim and Zino Lee, "Experience report on using a domain model-based extractive approach to software product line asset development", *Proceedings of the 11th International Conference on Software Reuse: Formal Foundations of Reuse and Domain Engineering*, Pages: 137-149, Springer, 2009.

[LI09] Dong Li and Carl K. Chang, ”Initiating and Institutionalizing Software Product Line Engineering: From Bottom-Up Approach to Top-Down Practice”, *33rd Annual IEEE International Computer Software and Applications Conference, COMPSAC '09*, Volume: 1, Pages: 53 – 60, IEEE Computer Society, 2009

[LIN00] Frank van der Linden and Henk Obbink, "ESAPS – Engineering Software Architectures, Processes, and Platforms for System Families", *Proceedings of the International Workshop on Software Architectures for Product Families*, Volume: 1951, Pages: 244-252, Springer, 2000.

[LIN02] Frank van der Linden, "Software Product Families in Europe: The Esaps & Café Projects," *IEEE Software*, Volume: 19, Number: 4, Pages: 41-49, 2002

[LIN04] Frank van der Linden, Jan Bosch, Erik Kamsties, Kari Känsälä and Henk Obbink, "Software Product Family Evaluation", *Software Product Lines*, Volume: 3154, Page: 110-129, Springer, 2004.

[LIN05] Frank van der Linden, "Family Evaluation Framework - overview & introduction", PH-0503-01, Information Technology for European Advancement and Philips Medical Systems, 2005.

[LIU06] Shih-Hsi Liu, Barrett R. Bryant, Jeff Gray, Rajeev Raje, Mihran Tuceryan, Andrew Olson and Mikhail Auguston, "QoSPL: A QoS-Driven Software Product Line Engineering Framework for Distributed Real-time and Embedded Systems", *Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering*, Pages: 724–729, 2006

[LUZ98] Robyn R. Lutz, Guy G. Helmer, Michelle M. Moseman, David E. Statezni and Stephen R. Tockey, "Safety analysis of requirements for a product family", *Third International Conference on Requirements Engineering*, Pages: 24, 1998.

[MAC96] Randall R. Macala, Lynn D. Stuckey Jr and David C. Gross, "Managing Domain-Specific, Product-Line Development ", *IEEE Software*, Volume 13 , Issue 3, Pages: 57 – 67, IEEE Computer Society, 1996

[MAN02a] Mike Mannion, "Using first-order logic for product line model validation", *Software Product Lines*, Pages: 149-202, Springer, 2002.

[MAN02b] Mike Mannion, "Organizing for Software Product Line Engineering", *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice*, Page: 55, IEEE Computer Society, 2002

[MAT02] Mari Matinlassi, Eila Niemelä and Liliana Dobrica, "Quality-driven architecture design and quality analysis method - A revolutionary initiation approach to a product line architecture", VTT Technical Research Centre of Finland, 2002.

[MAT03] Mari Matinlassi and Eila Niemelä, "The Impact of Maintainability on Component-based Software Systems", *Proceedings of the 29th Conference on EUROMICRO*, Page: 25, IEEE Computer Society, 2003

[MAT04a] Mari Matinlassi, "Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA," *26th International Conference on Software Engineering,* Pages: 127-136, 2004.

[MAT04b] Norihiko Matsuda, "Problems and Suggestion for Adopting Product Line Software Engineering from Modification Style Development", *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, Pages: 568 – 571, IEEE Computer Society, 2004

[MAT05] Gerardo Matturro and Andrés Silva, "A knowledge-based perspective for preparing the transition to a software product line approach", *Proceedings of the Software Product Line Conference*, Pages: 96-101, Springer 2005.

[MCG02] John D. McGregor, Linda M. Northrop, Salah Jarrad and Klaus Pohl, ”Initiating Software Product Lines”, *IEEE Software*, Volume 19, Issue 4, Pages: 24 – 27, IEEE Computer Society, 2002

[MEN08] Thilo Mende, Felix Beckwermert, Rainer Koschke and Gerald Meier, "Supporting the Grow-and-Prune Model in Software Product Lines Evolution Using Clone Detection", *Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering*, Pages: 163-172, IEEE Computer Society, 2008.

[MIL94] Matthew B. Miles and Michael Huberma, "Qualitative Data Analysis: An Expanded Sourcebook", Sage Publications Inc., 1994.

[MOO05] Mikyeong Moon, Keunhyuk Yeom and Heung Seok Chae, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line," *IEEE Transactions on Software Engineering,* Volume: 31, Number: 7, Pages: 551-569, IEEE Computer Society, 2005.

[MUL10] Gerrit Muller, ”Light weight architectures; the way of the future?”, http://www.gaudisite.nl/LightWeightArchitectingPaper.pdf, 2010, Accessed on 2010-08-03

[MUS08] Gunter Mussbacher, Daniel Amyot, João Araújo and Ana Moreira, "Modeling software product lines with AoURN ", *Proceedings of the 2008 Aspect Oriented Software Development workshop on Early aspects*, Article Number: 2, ACM, 2008

[NED07] Josef Nedstam and Mark Staples, “Evolving strategies for software architecture and reuse”, *Software Process Improvement and Practice*, Volume 12, Issue 3, Pages: 295-309, Wiley InterScience, 2007.

[NIE04] Eila Niemelä, Mari Matinlassi and Anne Taulavuori, ”Practical Evaluation of Software Product Family Architectures”, *Software Product Lines*, Volume 3154/2004, Pages: 164-168, Springer Berlin, 2004

[NÓB08] Jarley Palmeira Nóbrega, Eduardo Santana de Almeida and Sílvio Romero Lemos Meira, ”InCoME: Integrated Cost Model for Product Line Engineering”, *34th Euromicro Conference Software Engineering and Advanced Applications*, Pages:27-34, IEEE Computer Society, 2008

[NOO08] Muhammad Asim Noor, Paul Grünbacher and Christopher Hoyer, “A Collaborative Method for Reuse Potential Assessment in Reengineering-Based Product Line Adoption”, *Revised Selected Papers from the Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques*, Pages: 69-83, Springer Berlin/Heidelberg, 2008.

[NOR07] Linda Northrop, Paul Clements with Felix Bachmann, John Bergey, Gary Chastek, Sholom Cohen, Patrick Donohoe, Lawrence Jones, Robert Krut, Reed Little, John McGregor and Liam O'Brien, “A Framework for Software Product Line Practice, Version 5.0”, http://www.sei.cmu.edu/productlines/frame_report/, accessed 20100125, Software Engineering Institute, 2007.

[NUN09] Ingrid Nunes, Uirá Kulesza, Camila Nunes, Elder Cirilo and Carlos Lucena, ”Extending PASSI to model multi-agent systems product lines ”, *Proceedings of the 2009 ACM symposium on Applied Computing*, Pages: 729-730, ACM, 2009

[OBB00] Henk Obbink, Jürgen Müller, Pierre America and Rob van Ommering, "COPA: A Component-Oriented Platform Architecting Method for Families of

Software-Intensive Electronic Products", *The First Software Product Line Conference,* August 28–31, 2000, www.hitech-projects.com/SAE/COPA/COPA_Tutorial.pdf, accessed on 2010-08-20

[OMM02] Rob van Ommering and Jan Bosch, ”Widening the Scope of Software Product Lines — From Variation to Composition”, *Software Product Lines*, Volume 2379/2002, Pages31-52, Springer, 2002

[PAD05] Prasanna Padmanabhan and Robyn R. Lutz, ”Tool-supported verification of product line requirements”, *Automated Software Engineering*, Volume 12, Number 4, Pages447-465, Springer, 2005 .

[PAR05a] Shin Young Park and Soo Dong Kim, "A Systematic Method for Scoping Core Assets in Product Line Engineering", *Proceedings of the 12th Asia-Pacific Software Engineering Conference*, Pages: 491-498, IEEE Computer Society, 2005.

[PAR05b] Jaeil Park and Timothy W. Simpson, "Development of a production cost estimation framework to support product family design", *International Journal of Production Research*, Volume 43, Issue 4, Pages: 731-772, 2005.

[PET05] Ulf Pettersson and Stan Jarzabek, "Industrial experience with building a web portal product line using a lightweight, reactive approach”, *ACM SIGSOFT Software Engineering Notes*, Volume: 30, Issue: 5, Pages: 326 – 335, ACM, 2005

[PUR04] Anu Purhonen, Eila Niemelä and Mari Matinlassi, ”Viewpoints of DSP software and service architectures”, *Journal of Systems and Software*, Volume 69 , Issue 1-2 Pages: 57 – 73, 2004

[RAG98] Tara Ragan and Donald J. Reifer, "Adding Product Lines, Architectures, and Software Reuse to the Software Acquisition Capability Maturity Model", *The Journal of Defense Software Engineering*, Pages: 14-18, 1998.

[RIN98] David C. Rine and Robert M. Sonnemann, ”Investments in reusable software. A study of software reuse investment success factors ”, *Journal of Systems and Software*, Volume: 41, Issue:1, Pages: 17 – 32, Elsevier Science Inc, 1998

[RIV03] Claudio Riva and Christian Del Rosso, "Experiences with software product family evolution", *Proceedings of the 6th International Workshop on Principles of Software Evolution*, Page: 161, IEEE Computer Society, 2003.

[ROS09] Pedro O. Rossel, Daniel Perovich, and Marıa Cecilia Bastarrica, ”Feature Model to Product Architectures: Applying MDE to Software Product Line”, *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. WICSA/ECSA*, Pages: 201-210, 2009

[SAT07] Tonny Kurniadi Satyananda, Danhyung Lee, and Sungwon Kang, ”Formal Verification of Consistency between Feature Model and Software Architecture in Software Product Line”, *International Conference on Software Engineering Advances*, Page:10, IEEE Computer Society, 2007

[SAG00] Goiuria Sagarduy, Sergio Bandinelli and Ramón Lerchundi, “Product-line Analysis: Do we go ahead?”, *Proceedings of Workshop #15 at 22nd International Conference on Software Engineering*, Pages: 23-26, Fraunhofer IESE, 2000.

[SAV01] Juha Savolainen and Juha Kuusela, ”Violatility analysis framework for product lines ”, *Proceedings of the 2001 symposium on Software reusability: putting software reuse in context*, Pages: 133 – 141, ACM, 2001

[SAV07] Juha Savolainen Marjo Kauppinen and Tomi Mannisto, "Identifying Key Requirements for a New Product Line", *Proceedings of the 14th Asia-Pacific Software Engineering Conference*, Pages: 478-485, IEEE Computer Society, 2007

[SCH00] Klaus Schmid and Tanya Widen, ”Customizing the PuLSE^{TM} product line approach to the demands of an organization”, *Software Process Technology,* Pages: 221-238, Springer, 2000.

[SCH02a] Klaus Schmid and Isabel John, “Developing, Validating and Evolving an Approach to Product Line Benefit and Risk Assessment”, *Proceedings of the 28:th Euromicro Conference*, IEEE Computer Society, Page 272-283, 2002.

[SCH02b] Klaus Schmid, ”A comprehensive product line scoping approach and its validation”, *Proceedings of the 24th International Conference on Software Engineering*, Pages: 593-603, ACM, 2002.

[SCH02c] Klaus Schmid, ”The Product Line Mapping Approach to Defining and Structuring Product Portfolios”, *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, Pages: 219 – 226, IEEE Computer Society, 2002

[SEI10a] Software Engineering Institute, “CMMI Overview”, SEI at Carnegie Mellon University, 2010, http://www.sei.cmu.edu/CMMI/, accessed 2010-06-09.

[SEI10b] Software Engineering Institute, “Product Line Technical Probe”, http://www.sei.cmu.edu/productlines/consulting/techprobe/, Accessed 10-05-05.

[SHE09] Liwei Shen, Xin Peng and Wenyun Zhao, “A Comprehensive Feature-Oriented Traceability Model for Software Product Line Development”, *Proceedings of the Australian Software Engineering Conference of 2009*, Pages: 210-219, IEEE, 2009.

[SHI06] Suk Kyung Shin, Jin Sun Her and Soo Dong Kim, "Applying Formal Approach to Core Asset Instantiation in Product Line Engineering", *Proceedings of the 13th Asia Pacific Software Engineering Conference*, Pages: 427-434, 2006.

[SIE08] Norbert Siegmund, Martin Kuhlemann, Marko Rosenm¨uller, Christian Kaestner, and Gunter Saake, ”Integrated Product Line Model for Semi-Automated Product Derivation Using Non-Functional Properties”, *Proceedings of the Workshop on Variability Modelling of Software-intensive Systems,* ICB Research Report, Pages 25–31, University of Duisburg-Essen, 2008.

[SIM95] Mark A. Simos, ”Organization domain modeling (ODM): formalizing the core domain modeling life cycle”, *Proceedings of the 1995 Symposium on Software reusability*, Pages: 196 – 205, ACM, 1995.

[SIM96] Mark Simos, Dick Creps, Carol Klingler, Larry Levine and Dean Allemang, ”Software Technology for Adaptable Reliable Systems (STARS) Organization Domain Modeling (ODM) Guidebook Version 2.0”, STARS-VCA025/001/00, Manassas, VA, Lockheed Martin Tactical Defense Systems, 1996.

[SIM02] Daniel Simon and Thomas Eisenbarth, “Evolutionary Introduction of Software Product Lines”, *Proceedings of the Second International Conference on Software Product Lines*, Page: 272–283, Springer-Verlag, 2002.

[SNO08] Colin Snook, Michael Poppleton and Ian Johnson, ”Rigorous engineering of product-line requirements A case study in failure management”, *Information and Software Technology*, Volume 50, Issues 1-2, Pages 112-129, Elsevier, 2008.

[SOC06] Periklis Sochos, Matthias Riebisch and Ilka Philippow, ”The Feature-Architecture Mapping (FArM) Method for Feature-Oriented Development of Software Product Lines”, *Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, Pages: 308 – 318, IEEE Computer Society, 2006

[SOM05] Marius Sommerseth, ”Reuse through product-families and frameworks”, *TDT4735 Software Engineering, Depth Study*, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science (IDI), 2005

[STA04] Mark Staples and Derrick Hill, Experiences adopting software product line development without a product line architecture, *Proceedings of the 11th Asia-Pacific Software Engineering Conference,* Pages: 176-183, IEEE Computer Society, 2004.

[STE04] Mirjam Steger, Christian Tischer, Birgit Boss, Andreas Müller, Oliver Pertler, Wolfgang Stolz and Stefan Ferber, “Introducing PLA at Bosch Gasoline Systems experiences and practices”, *Proceedings of the International Software Product Line Conference*, Pages: 34-50, Springer, 2004.

[STO02] Christoph Stoermer and Markus Roeddiger, “Introducing product lines in small embedded systems”, *Proceeding of the Fourth International Workshop on Software Product-Family Engineering*, Page: 101-112, Springer, 2002.

[SUN08] Jong Sung Dong, Keun Lee, Kyong Hwan Kim, Sang Tae Kim, Ji Man Cho, Te Hi Kim, ”Platform maintenance process for software quality assurance in product line”, *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*,Volume: 2, Pages: 325-331, IEEE Computer Society, 2008.

[SVA02] Mikael Svahnberg and Michael Mattsson, “Conditions and restrictions for product line generation migration”, *LNCS 2290*, Page: 143–154, Springer-Verlag 2002.

[THI01] Steffen Thiel, Stefan Ferber, Thomas Fischer, Andreas Hein and Michael Schlick, “A Case Study in Applying a Product Line Approach for Car Periphery Supervision Systems”, *Proceedings of In-Vehicle Software 2001,* Page 43-55, Society of Automotive Engineers, Inc., 2001.

[THI02] Steffen Thiel, “On the Definition of a Framework for an Architecting Process Supporting Product Family Development”, *Proceedings of the 4th International Workshop on Product Family Engineering*, Volume: 2290, Pages: 3-47, Springer, 2002.

[THO03] Jeffrey M. Thompson and Mats P. E. Heimdahl, ”Structuring product family requirements for n-dimensional and hierarchical product lines”, *Requirements Engineering*, Volume 8, Number 1, Pages: 42-54, Springar London, 2003

[TIS07] Christian Tischer, Andreas Müller, Markus Ketterer and Lars Geyer, ”Why does it take that long? - Establishing product lines in the automotive domain”, *Proceedings of the 11th International Software Product Line Conference,* Page 269-274, IEEE Computer Society, 2007.

[TOM02] Amir Tomer and Stephen R. Schach, "A three-dimensional model for system design evolution", *Systems Engineering*, Volume: 5, Issue: 4, Pages: 264-273, Wiley Periodicals Inc. 2002.

[TRA06] Bruce Trask, Dominick Paniscotti, Angel Roman and Vikram Bhanot, ”Using model-driven engineering to complement software product line engineering in developing software defined radio components and applications” *Conference on Object Oriented Programming Systems Languages and Applications. Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications,* Pages: 846 – 853, ACM, 2006

[TUN08] Emanuel Tundrea, ”SmartModels an MDE platform for the management of software product lines”, *Proceedings of the 2008 IEEE International Conference on Automation, Quality and Testing, Robotics - Volume 03*, Pages: 193-199, IEEE Computer Society, 2008

[YE05] Huilin Ye and Hanchang Liu, "Approach to Modeling Feature Variability and Dependencies in Software Product Lines", *IEE Proceedings in Software of 2005*, Volume: 152, Issue: 3, Pages: 101-109, IEE, 2005.

[YOS06] Kentaro Yoshimura, Dharmalingam Ganesan and Dirk Muthig, "Defining a strategy to introduce a software product line using existing embedded systems", *Proceedings of the 6th ACM & IEEE International conference on Embedded software*, Pages: 63-72, ACM, 2006.

[VEH00] Tuomo Vehkomäki and Kari Känsälä, “A Comparison of Software Product Family Process Frameworks”, Springer-Verlag, 2000.

[VOE07] Markus Voelter and Iris Groher, ”Product Line Implementation using Aspect-Oriented and Model-Driven Software Development”, *Proceedings of the 11th International Software Product Line Conference*, Pages: 233-242, IEEE Computer Society, 2007

[VOG02] Stefan Voget and Martin Becker, "Establishing a Software Product Line in an Immature Domain", *Proceedings of the 2nd International Conference on Software Product Lines*, Pages: 60-67, Springer, 2002.

[WIJ02] Jan Gerben Wijnstra, "Critical Factors for a Successful Platform-Based Product Family Approach", *Proceedings of the 2nd International Software Product Line Conference*, Volume: 2379, Pages: 15-35, Springer, 2002.

[WIT09] Paul D. Witman, ”Software Product Lines and Configurable Product Bases in Business Applications - A Case from Financial Services”, *42nd Hawaii International Conference on System Sciences, HICSS '09*, IEEE Computer Society, 2009

[ZHA06] Weishan Zhang and Thomas Kunz, “A Product Line Enhanced Unified Process”, *Proceedings of the International Software Process Workshop and International Workshop on Software Process Simulation and Modeling 2006,* Page: 142–149, Springer-Verlag, 2006.

[ZHU08] Jun Zhu, Quan Yin and Rui Zhu, "A Plugin-Based Software Production LineIntegrated Framework", *Proceedings of the International Conference on Computer Science and Software Engineering*, IEEE Computer Society, 2008.

7 Appendix A: Review Protocol

Background

Rationale for the survey is twofold. Firstly there are a lot of presented works on SPL but no literature review have been presented that covers and summarizes these works. From our literature review we want to compile a manageable collection of articles and practices that companies can use as a base for continued investigation of SPLs. Secondly, the result of the systematic literature review will be used to construct guidelines on how to adopt SPL development. These guidelines will mainly be based on experience reports and lessons learned.

The research questions

- RQ1. How should the SPL practices be structured?
  - RQ1.1. What available frameworks for SPL development are there?
  - RQ1.2. What are the differences between the found frameworks?
  - RQ1.3. Which of these can be recommended?

- RQ2. What methods should an organization consider for their SPL development?
  - RQ2.1. Which are the specific areas of software product lines?
  - RQ2.2. What available methods are there for each area?
  - RQ2.3. Which of these can be recommended?

- RQ3. How does an organization adopt a SPL approach?
  - RQ3.1. What available adoption approaches are there?
  - RQ3.2. Are there any differences between the adoption approaches proposed by researchers and those used in industry?
  - RQ3.3. What are the differences between specific adoption approaches?
  - RQ3.4. What variables affect the choice of adoption approach?
    - RQ3.5. What can be learned from industry case studies?
  - RQ3.6. Which of the found adoption approaches can be recommended?

- RQ4. What are the available methods on analysing an organization's readiness towards adopting a SPL approach?

- RQ5. What are the consequences of combining SPL development and CMMI process improvement?
  - RQ5.1: Are there any similar procedures or concepts between the two initiatives?
  - RQ5.2. Does the CMMI maturity levels affect the SPL adoption process?
  - RQ5.3. Are there any CMMI process areas that have a greater benefit for SPL development than for single system development?
  - RQ5.4. Are there any CMMI process areas that are more difficult to apply in SPL or has a negative effect on SPL development?

- RQ5.5. Can the introduction or improvement of SPL practices improve the organization's process maturity as specified by CMMI?

- RQ6. Can SPL practices be categorized into maturity levels?

Primary search strategy

The literature reviews will be conducted on three resource databases and be limited to the last decade (2000-2010). The time frame was chosen with the rationale that the SPL research area is fairly new and that any older works of importance would be well referenced in the literature we would find. Because of this the reference lists of the selected articles were manually scanned for articles of relevance. Manual scanning of conference proceedings was deemed unnecessary since the test runs of our chosen databases covered all relevant conference proceedings, symposiums and journals.

Our choices of databases is based on our database review and we will use Compendex combined with Inspec through the engineering village search engine, Google Scholar and CiteSeerX. Google Scholar is familiar to us and has a wide search pattern that includes alternative sources. When combining Inspec and Compendex you get a thorough search of scientific journals and conference proceedings. Finally we chose CiteSeerX as our third database since it's main focus is Computer Science. Ebrary and Libris is excluded because they are covered by the other databases. Web of Science is similar to Inspec and Compendex except that it does not include conferences, which make it unnecessary to include. Scopus database is containing articles from the medicine and biology areas and is excluded as well.

Our choices correspond with those of Libraries of MIT:
http://libstaff.mit.edu/refcomm/training/2008eslcrosstraining.rtf

Database review:

- Google scholar: Easy to use and are continuously expanding. No required registration and a well known database.
- Inspec contains major journals, articles and conference proceedings from various engineering disciplines.
- Compendex contains titles from most major engineering disciplines.
- Libris, searching all available titles from Swedish libraries
- Ebrary, containing many library titles.
- Web of Science, is searching through a broad base of science journals.
- Scopus, its database consists primarily of medicine, biological and environmental science articles.

CiteSeerX, Is public and consists primarily of computer science.

Our search is divided on three search strings from which the results will then be merged. The search terms used in the search strings are divided on four groups, see Table 1. The search strings will be constructed with boolean AND between groups of search terms and boolean OR between search terms within a group. The three searches will be constructed according to Table 2.

Table 7: Groups of search terms

Group 1	Software product line/lines, software product family/families, software product factory/factories.
Group 2	framework(s), model(s), practice(s)
Group 3	maturity, readiness, preparedness.
Group 4	initiation, adoption, introducing.

Table 8: Search string construction

Search 1	At least one term of group one and at least one term of group two
Search 2	At least one term of group one and at least one term of group three
Search 3	At least one term of group one and at least one term of group four.

Study selection criteria and procedures.

The following table shows the Inclusion and Exclusion criteria for the systematic literature review. When a criteria is true, the article is included or excluded and the rest of the criteria is ignored.

Table 9: Inclusion/Exclusion criteria

Inclusion/Exclusion Criteria	If true
Not written in English.	Exclude
Redundant (there exists later versions of the article etc.)	Exclude
The subject is not concerning SE and SPL.	Exclude
Concludes an artefact as a SPL success factor.	Include
Includes lessons learned or suggestions based on case study.	Include
Includes comparison with or references to CMMI	Include
Discusses a model, method or framework for SPL.	Include
The subject is concerning SPL maturity, readiness or categorizing of SPL organizations	Include
Discuss a method or approach for initiating SPL.	Include
Not well referenced (empty statements)	Exclude

The initial database searches will identify a number of papers, the titles and abstract of these will be read and compared against the study exclusion criteria. Full papers of those articles that could not be excluded will be obtained. The reference lists of the selected articles will be manually scanned and articles of relevance will be obtained. Once these copies are obtained the full texts will be read. The articles will then be compared to the inclusion/exclusion checklist one more time.

Study quality assessment checklists and procedures.

The quality of the studies will be categorized as satisfactory or unsatisfactory. The assessment will be made during discussion of the articles. A satisfactory articles should contain at least one of the following attributes.

- A clear definition of at least one model or framework for SPL.
- A scientific and well performed case study on a organization using a certain model or framework for SPL.
- A clear definition on SPL maturity/readiness.
- A clear definition on maturity of artefacts of a SPL.
- A description of how to categorize SPL in a novel way.
- A clear definition on at least one approach for initiating SPL.
- A scientific and well performed case study on a organization which adopted the SPL approach.

Data extraction strategy.

All articles obtained will be reviewed and summarized by reading the articles followed by a discussion on each article. The summaries will be written in plain text. Based on the discussion of the articles they will be categorized in appropriate categories for easy access. Then when working on a specific area the summary of those articles will be read and when needed the article will be read again to clarify any issues.

Synthesis of the extracted data.

- The frameworks will be fully evaluated and listed with benefits, drawbacks, similarities, differences and a grade of usefulness.
- The maturity articles of SPL and organization will be used to in an attempt to categorize adoption readiness.
- Adoption approaches will be evaluated and divided into categories depending on type of adoption approach.
- Experience reports and lessons learned will be used to construct guidelines on how to start a SPL adoption initiative.

Preliminary project timetable

Preparation of literature reviews	10-02-15	10-02-21
SLR on SPL models	10-02-22	10-03-14
SLR on SPL maturity	10-03-15	10-03-21
SLR on SPL adoption	10-03-22	10-03-28
SLR result documentation	10-03-29	10-04-04

8 APPENDIX B: FRAMEWORK SUMMARY

Framework: Framework of Software Product Line Practices (Henceforth the SEI Framework) [NOR07]

Authors: Software Engineering Institute (SEI)

Description: Identifies and listing the fundamental concepts and activities to adopt the software product line approach, including knowledge needed and issues that need to be considered. Focusing on three areas, Core Asset Development, Product Development and Management. These exists in concert with each other, either assets are created from old products or products are created from new assets, both under both technical and organizational management. These three essential activities is a must for successful SPL development, to be able to carry out these you must master a collection of practice areas. These practice areas are organized in to three groups, Software Engineering, Technical Management and Organizational Management. Software Engineering practice areas are handling architecture definition and evaluation, asset creation and extracting, requirement engineering, domain analysis, and finally testing and integration. Technical Management practices are the management practices that must me mastered to be able to develop and change both core assets and products. More precise it contains; configuration management, commission analysis, scoping, measurement and tracking, technical planning, technical risk management and tool support. Organizational management practices are practices handling the running of the entire product line effort. Business case, interface management, acquisition strategy, funding, institutionalizing, organizational planning, structuring and risk management, training and finally operations.

Evaluation: Very thorough framework created in the late 90's and has evolved since. Authors, co-authors and previous authors are many and well known within the SPL areas. Well known and indirectly used, for example as a base for new frameworks.

Framework: Business Architecture Process and Organization (BAPO) [LIN02, LIN04] and Family Mature Framework (FEF) [LIN05]

Authors: Frank van der Linden, Jan Bosch, Erik Kamsties, Kari Känsälä and Henk Obbink

Description: The BAPO acronym was first mentioned by Henk Obbink and consists of four dimensions, Business, Architecture, Process and Organization. Different variations of a framework using the BAPO dimensions has been presented in several articles [LIN02, LIN04, LIN05]. The latest and the most extensive of these are the Family Evaluation Framework (FEF) [LIN05]. The FEF provides evaluation on each of the four BAPO concerns separately which gives insight where an organization can improve.

BAPO is and early version of the FMF and both are evaluation frameworks that thoroughly explains all dimensions and all phases, steps and levels inside the dimensions. In business there are 5 maturity levels, Reactive, Awareness, Extrapolate, Proactive and Strategic. The architecture is also divided into 5 groups, Independent product development, Standardized infrastructure, Software platform, Variant products and Self-configurable products. The organization is divided into aspects, Unit oriented, Business lines oriented, Business group/division, Inter-division/companies and Open business. The Process concern uses the maturity levels from CMMI, Initial,

Repeatable, Defined, Managed and Optimized. This is covered in a separate framework, the Family Maturity Framework (FMF) [KÄN05].

Evaluation: Using this framework gives you a good overview of your entire SPL effort. If one part has lower maturity then the other, then investigating why can lead to findings of issues or things overlooked. Well referenced.

Framework: Reuse-driven SW Engineering Business (RSEB) [JAC97].

Author: Ivar Jacobson, Martin Griss and Patrik Jonsson

Description: RSEB is a systematic model for implementing reuse and uses object-oriented analysis and design, UML and layered software architecture. There are clear instructions on how to perform the analysis, the design, the implementation and the validation. There are not any clear instructions on how to perform the management though. The process is constructed with separate activities in three major categories, Application Family Engineering, Component System Engineering and Application System Engineering. Application Family Engineering is to develop a conceptual model and a common layered architecture for all product line members. Component System Engineering is to develop functional building blocks for the layered product platform. Application System Engineering is to assemble the products with help of the architecture and building blocks created in the other two categories.

In addition to this engineering activities, there is also a set of activities that support the transition to a reuse approach.

Evaluation: A good framework which is a bit outdated. There is however a newer version of this framework where they integrate it with FODA and UML (FeatuRSEB) but it still focuses on features and domain modelling and lacks on management parts.

Framework: Generic Product Line Process Framework (The Generic Framework) [VEH00]

Author: Nokia Research Center (NRC)

Description: Framework based on other frameworks like the SEI Framework and RSEB. It consists of these Activity Groups which are divided into four major groups, Product Line Engineering, Application Engineering, Domain Engineering and Third Party Product Acquisition and Subcontracting. The Generic Framework creates systems that are composed of four layers; system, product, platform, and component. Component are building blocks to upper layers. Platform consists of several components. Products consists of platforms and/or components integrated together. Finally, systems are complete solutions that consist of several products together.

Evaluation: A quite brief but good framework. The activity groups are described briefly and what activities they contain. Well referenced and contains the best attributes of the four frameworks it is based on.

Framework: Product Line Engineering Practices Model (The PLEP Model) [COA05]

Author: François Coallier

Description: This framework consists of 31 Product Line practice areas, grouped in five categories. An important objective of the PLEP model is that it should be easily incorporated into existing development methodologies and remaining aligned with existing systems engineering standards. It is based on the SEI framework and the VRAPS model but has a stronger emphasis on system considerations, where the SEI framework and VRAPS mostly concentrate on software. The five categories of

practice areas are; Product Line Management, Architecture Management, Requirements Management, Assets Development and Management and finally Product Synthesis and Support. Instead of using skills to define the categories like SEI, they have grouped their practice areas based on activities to remain consistent with the CMMI and the approaches currently used in the software and system engineering capability areas. The VRAPS model is about creating the architecture with a clear Vision, a good Rhythm, well Anticipation, uncomplicated Partnering with stakeholders and Simplification of environment. These five guidelines are incorporated into the PLEP Model.

Evaluation: A well referenced and well structured framework with well described practice areas and belonging purpose. It is also based on one well known framework and one well used model.

Framework: The Integrated Process (TIP) [KIM06a]

Author: Soo Dong Kim, Hyun Gi Min, Jin Sun Her and Soo Ho Chang

Description: An integration between Component Based Development (CBD), Model Driven Architecture (MDA) and SPL. Integrating these three development areas supports the three key criteria of TIP, Reusability, Productivity and Standardization. The Integrated Process consists of two main parts with subareas, Core Asset Engineering and Application Engineering. Core Asset Engineering has Domain Analysis, Product Line Scoping Component Acquisition and Core Asset Modeling as its subareas. Application Engineering has Application Requirement Analysis, Define Decision Resolution Model, Application Specific Design, Core Asset Instantiation, Model Integration, Application Detailed Design, Component Customization, Application Implementation and Application Integration.

Evaluation: A good framework which might not be so extreme and novel as they claim. It includes much on how to create the specific product out of the core assets though. Describing what to do and including many specific methods and tools which tells you how to do it as well.

Framework: Yet Another Make (YAM) [JAI06]

Author: Abhinandan Jain and Jeffrey Biesiadecki

Description: YAM is a framework with belonging tools for rapid development of software in a fitting environment. YAM divides the activities and solutions based on five groups of needs. Basic Development Needs, which is the lowest level of needs needed for functional development. Secondly there is the Stable Concurrent Development Needs which are needs necessary for when there is more then one developer and thus there is processes to manage and coordinate. Next level is the High Quality Software Needs, including needs for performance, robustness, maintainability and defect rates. Fourth category of needs are the Rapid Development Needs which is the need for increasing the team's productivity and development paces. And finally, the Reuse Needs, these are the highest level of needs and enables reusing to create multiple products from same components like in a SPL.

Evaluation: This framework is not specifically for SPL. It is a good framework though, with a innovative way to group the activities needed for SPL.

Framework: BigLever's Software Product Line Lifecycle Framework (BigLever's Framework) [BIG09]

Author: BigLever's Software Inc.

Description: Allows the SPL development process to flow easily from phase to phase by eliminating unnecessary steps and intermediary and providing common SPL concepts, constructs, tools and methods throughout the whole lifecycle for all involved.

Evaluation: This framework is not publicly available and can thus not be fully evaluated. It is a used framework though and works well according to the creators and users. Table 8: Framework Comparison

Framework	Source	Validation	Provided Assistance	Version	Comment	Practice Areas
The SEI Framework	[SEI07]	Medium High	Illustrations and tool support	Optimized	No additional comment	Complete
The BAPO Framework	[LIN04]	Medium	None specific	Evolved	Used for evaluation of a SPL	Not applicable
The RSEB Model	[JAC97]	Medium	None specific	Evolved	Outdated	Lacks management
The Generic Framework	[VEH00]	Low	None specific	Initial	Based on FSPLP, RSEB SPICE and DsE	Complete
The PLEP Model	[COA05]	Low	VRAPS Guidelines	Initial	Based on SEI and VRAPS	Complete
The Integrated Process	[KIM06]	None	Method and tool support	Initial	No additional comment	Complete
Yet Another Make	[JAI06]	Medium	None specific	Nearly Optimized	Not specifically for SPL	Lacks in SPL processes
The BigLever Framework	[BIG09]	Medium High	Tool support	Nearly Optimized	Not publicly available	Unknown

Validation: *None, Low – Based on existing models or frameworks only, Medium Low – Internal case study and example provided and/or mentioned and used by others, Medium – Well known, used and/or used as a basis for other frameworks, Medium High – Used in industry, High – Used in industry with quantitative and qualitative data provided.*

Provided Assistance: *Can be in form of examples, methods, tools or guidelines.*

Version: *Initial – First draft of framework, Evolved - Evolved once or twice after first draft, Nearly Optimized – Evolved several times and updated based on cases, Optimized – Evolved multiple times and optimized based on cases and lots of experience, Finalized – Final version.*

Comment: *Additional comments on the framework, often some flaws.*

9 APPENDIX C: METHOD SUMMARY

Domain analysis methods

Method: Feature-Oriented Domain Analysis (FODA) [KAN90]
Author: Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S
Description: The most referenced method for feature modeling is FODA, possibly because it was one of the first methods to gain public acknowledgement and support and thus many other methods extend on, or compares itself with, FODA. The sole purpose of FODA is to model features, their relations and dependencies. FODA was not originally intended to be used to model complex SPLs and is thus simple and rather basic. However as mentioned there are many methods that extend FODA and allow more complex relationships and consider the challenges of reuse. FODA is comparatively simple and fast to use and could be recommended for use when no complex relationships are needed. This is however not the usual case when modeling SPLs.
Evaluation: An early feasibility study of FODA showed its potential for domain analysis [KAN90]. The study was conducted before SPL engineering and systematic reuse had become the big thing so this aspect was not really considered in FODAs design. FODA is well referenced by many authors and articles in later years. Most agree that FODA suits the purpose it was designed for but also point out that modern SPL engineering often requires a more complex representation. Many who point this out proposes extensions to FODA as the solution. One case study that uses concepts from FODA is [JEN07].

Method: Feature-Oriented Reuse Method (FORM) [KAN98]
Author: Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Gerard Jounghyun Kim, Euiseob Shin
Description: FORM is one of the more well referenced extensions to FODA with the aim of specifying reusable assets and architecture components. The strength of FORM is that it has kept much of the simplistic notation of FODA while adding additional viewpoints to make it suitable to model SPL reusability. Where FODA focuses only on the features and their relations, FORM goes a step further and considers the design of reusable architectures and the mapping between reusable assets and a reference architecture.
Evaluation: FORM is a feature modeling method that should be well suited for SPL engineering. The authors claim successful application in industry although it is hard to otherwise tell how well used the method is in industry since experience reports and case studies usually state that they use feature modeling and do not elaborate on the method or notation.

Method: Organization Domain Modeling (ODM) [SCH02b]
Author: Mark A. Simos
Description: While FODA has a rather narrow focus on features and their modeling and FORM takes the step toward design, ODM takes one step backwards in the chain of activities and includes explicit scoping activities. Still, the ODM method

takes a clear feature viewpoint and is mainly a modeling method. Among the strengths of ODM is an extensive guidebook for the practical use of the method [SIM96].

Evaluation: ODM has been applied in several private as well as military projects. One of the projects where ODM was one of the methods used was awarded with a Federal Technology Leadership Award. The fact that ODM has been repeatedly used in military projects and contributed to a project being awarded would make ODM one of the most validated methods to be found in this research field, even though there are no quantitative evidence to support claims of success.

Method: Featured Reuse-driven Software Engineering Business (FeatuRSEB) [GRI98]

Author: Griss M., Favaro J., d'Alessandro M

Description: FeatuRSEB combines the strengths of FODA's feature models with use case modeling from the Reuse-Driven Software Engineering Business (RSEB) method. The combination of feature modeling and use case modeling seems to be effective and FeatuRSEB is one of the methods that make use of this advantage. In FeatuRSEB the feature model is in the center and the use case model is a complement.

Evaluation: The authors claim successful application in industry although no facts are presented to support the claim. We have also been unable to find any case study or experience report that can provide these facts. However, both legacy methods, FODA and RSEB, are well represented in literature and the combination represented by FeatuRSEB should give a robust method.

Method: A two view feature modeling method [FER02]

Author: Stefan Ferber, Jürgen Haag, and Juha Savolainen

Description: The authors of this method suggest the use of multiple views on feature models, much like it is used for architecture design. This method uses a FODA like notation for one of the views and suggest syntax and semantics for an interaction and dependency view. The multiple views effectively increases the understanding of the model as a whole and eases the readability of each individual view. The concept is easily understandable and some advantages are apparent, but the article lacks somewhat in completion and validation.

Evaluation: This is one of quite a few papers that suggest multi view feature models and there are apparent advantages with such models. The method presented in this paper does not make any claim of validation and has the feel of a work in progress.

Method: A multi view feature modeling method [FEY02]

Author: Dániel Fey, Róbert Fajta, and András Boros

Description: Like [FER02], this method uses a multi view approach to feature modeling. However this paper defines a more complete set of notation and provides more examples of how to use it which make this paper more pedagogical and easy to follow.

Evaluation: The paper in itself is filled with examples and illustrations that should make the learning of the method easy. The method was tested in a pilot project which the authors claims had good results. We find no evidence that the method has been used in industry.

Method: Product Line Use case modeling for System and Software engineering (PLUSS) [ERI05]

Author: Magnus Eriksson, Jürgen Börstler, and Kjell Borg

Description: Similar to FeatuRSEB, PLUSS uses a combination of feature modeling and use case modeling. As explained in [ERI05] the fundamental difference between FeatuRSEB and PLUSS is that while FeatuRSEB focuses on the feature model, PLUSS puts the most importance on the use case model. The use case modeling of PLUSS builds on the legacy of Rational Unified Process (RUP), while the feature modeling is a modification of the FODA notation. Tool support for PLUSS includes Telelogic DOORS and IBM-Rational Rose.

Evaluation: The PLUSS approach as been applied in the Swedish defense industry with reported good results [ERI06]. Since the company in the experience report was already familiar with RUP and the tools used they found the transition to the PLUSS approach relatively easy [ERI06]. Tool support is often lacking for many domain engineering methods. To make use of commercially available tools such as DOORS and Rose as support can be considered a big advantage.

Method: An Approach to Developing Domain Requirements [MOO05]

Author: Mikyeong Moon, Keunhyuk Yeom

Description: In this article a method is proposed to express product line requirements in such a way that product requirements can be derived as a subset of the product line requirements. The method is centered around matrix analysis of commonality and variability between the individual products. The commonality and variability is expressed as mandatory or optional requirements and is complimented with actor use case modeling to express valid interactions. The authors also provide a specially designed tool named DREAM to aid the user of the method.

Evaluation: Matrix analysis tend to be a very time and effort consuming method to use as system complexity is scaled up. How well the tool DREAM works to easy this workload is hard to estimate. The method is validated with a case study in cooperation with a research institute and has as far as we know not been used in industry.

Method: Product-Line Requirements Specification (PRS) [FAU01]

Author: Stuart R. Faulk

Description: Suggest a requirement specification with alternative and optional requirements for the SPL where the requirements for each product derived from the SPL can be specified by a subset of the generic requirements. The paper explains the concepts and gives examples on how it can be done.

Evaluation: Interesting and inspiring paper. It is still a work in progress as the authors themselves point out. The method is only partly validated in a case study and we have found no experience reports concerning this or similar methods. It is thous hard to evaluate the benefits of methods such as this one.

Method: Using first-order logic for product line model validation [MAN02a]

Author: Mike Mannion

Description: The approach taken by this method is influenced by the feature modeling methods. The requirements are categorized as mandatory domain requirements, mutual exclusion, multiple choice and optional requirements. These requirements are then organized into a parent-child relation three structure, much like

classical feature modelling. The difference is that with this method this is done on the requirement level instead of the feature level. This method also proposes a support for validation of derived product specifications by the use of rather simple first order logic.

Evaluation: The advantages of feature modelling on the requirement level rather than the feature level of abstraction is rather unclear. The automatic validation of a derived system using first order logic on the other hand seems to have its strong points. The validation gained by their examples are however quite shallow and does not make system and integration testing any less important. The method is demonstrated in the paper but there are no claims of industry validation.

Architecting and Design methods

Method: Company Oriented Platform Architecting (COPA) [OBB00]

Author: Henk Obbink, Jürgen Müller, Pierre America, Rob van Ommering, Gerrit Muller, William van der Sterren and Jan Gerben Wijnstra

Description: COPA incorporate more than the architecture creation as the method is heavily influenced by BAPO and cover all the aspects of product line engineering i.e. architecture, process, business and organization. However the architecture plays a very central role in the method and the architecture creation is concentrated on balancing between top-down and bottom-up approaches. COPA is focused on the creation of a SPL, There is very little mention of the SPL usage.

Evaluation: COPA has been applied and reported in several case studies made at Philips Research. Overall it is a common approach for SPL creation. A reason for this is possibly that COPA was quite early and described what SPL was about before many other works had been presented. Another reason is that other works likely have been influenced by COPA.

Method: Family-Oriented Abstraction, Specification and Translation process (FAST) [Software product-line engineering: a family-based software development process]

Author: David M. Weiss and Chi Tau Robert Lai

Description: The FAST method defines a whole process for product line engineering, including domain engineering and production. It describes activities, artefacts and roles. As it is a family oriented method it emphasise the architectures role in the development, and the creation of the architecture is an important part of the method.

Evaluation: According to [MAT04a] FAST is very adaptable but not applicable as it is. The method has been developed and applied in industry at least at Lucent Technologies.

Method: KobrA [ATK00]

Authors: Colin Atkinson, Joachim Bayer, and Dirk Muthig

Description: The KobrA project was funded by the Fraunhofer institute and the method is described as a ready to use customization of the PuLSE methodology. Thus it includes the entire SPL development. Compared to the PuLSE methodology the KobrA method is located at a lower abstraction level and is much more hand on. The method is well described and includes many examples. To complement the main

article there is also a very extensive case study of la library system SPL where the KobrA method has been used [BAY01]. The case study includes much more detail than is normal and the processes and resulting artefacts are easy to follow and understand.

Evaluation: [MAT04a] describe KobrA as: "Practical, simple method for traditional component-based software engineering with UML. Adapts to both single systems and family development." The method is fairly simple and as it builds on PuLSE it incorporate all the most essential SPL concepts. The rich descriptions and the extensive case study does make the method easier to understand and use than many other methods. When it comes to validity there are not much claim for it other than that one case study [BAY01]. However since the method is based on and very similar to PuLSE it inherits some validity.

Method: A Systematic Process to Design Product Line Architecture [KIM05a]

Author: Soo Dong Kim, Soo Ho Chang, and Hyun Jung La

Description: Proposes a method, including guidelines and instructions, based on common practices found in other methods. The introduction section gives a good overview of what a Product Line Architecture (PLA) is and how it is used. The method in itself is structured and intuitive.

Evaluation: Unfortunately the method has not been validated at all. It is not a novel approach and comparable methods has been somewhat validated to work. What is special with this paper and what made us include it is the guidelines and instructions given for all activities. The amount of instructions given in this article is very unusual, which makes it hard to make most methods repeatable. We would like to see instructions included in more methods and not just the concepts behind it.

Method: Quality-Driven System Architecting of product families (QUASAR) [THI02]

Author: Steffen Thiel

Description: A process framework for SPL architecting. The paper describes a suggested workflow to derive an architecture. The activities are divided into three main groups, preparation, modeling and evaluation. There is no or little help on how to do the actual activities suggested in the workflow. A stable scope is identified as a prerequisite for the method.

Evaluation: A workflow model without instructions on the activities can indicate a highly customizable method. The paper does not claim any validation.

Method: Extended ATAM (EATAM) [KIM08]

Author: Taeho Kim, In-young Ko, Sung-won Kang , Dan-hyung Lee

Description: Many methods for creating SPL architectures suggest that the architecture should be evaluated with some previously well known method used in single system development, such as ATAM. The problem with these methods, as often stated by the same people who suggest them is that they offer little support for generic product line architecture complexities. One of the attempts to correct this problem is presented as this method called EATAM.

Evaluation: EATAM is the only systematic method for this purpose we have found to be at least claimed to work. The paper presents a case study of EATAM applied in the electrical appliances field with good results. Methods like EATAM would probably be considered labor intensive compared to the single system

counterparts but the time spent during architecture validation should also be considered a well made investment since the architecture plays such a vital role in SPL engineering.

10 APPENDIX D: THE PULSE METHODOLOGY

Authors: Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen and Jean-Marc DeBaud at the Fraunhofer Institute for Experimental Software Engineering.

Description: PuLSE is a process framework for practice area specific methods tied to the PuLSE methodology. Several methods can be used more or less on its own, but they are designed to complement each other. The work output from one method will be used as input to the next method in the chain, and they can be governed by management and control processes that are also part of the PuLSE methodology. The methods are focused on the work processes and in what order to execute them. PuLSE has been designed to be highly customizable [DEB98a], and by choosing what practices to include and customize it should fit almost any organization of any size. A special study has even been made on how the methodology works in small and medium sized companies showing positive results[KNA00]. There is also a specially developed tool belonging to the methodology called DIVERSITY/CDA to help the user in some of the more work intensive activities of SPL engineering [BAY99a]. This speaks well for PuLSE since tool support is often lacking in the SPL field.

The PuLSE methodology is structured around four deployment phases. To each deployment phase belongs on or more of six technical components and its all supported by three support components [BAY99a, DEB98a]. In this paper we will focus on the technical components, which are briefly described below. Another method closely related to PuLSE is KobrA that is described as a ready to use instantiation of PuLSE. KobrA is described in Appendix C: Method Summary.

PuLSE-BC

The PuLSE-BC (Baselining and Customization) component is used by the PuLSE methodology internally to customize the PuLSE processes before application in a project. This is the PuLSE methodology's answer to the problem statement that no company or project is the same [SCH00]. The explicit identification of how the processes can vary, validation of these variations and customization of the processes to each project makes the PuLSE methodology very flexible and applicable in companies and projects of different size and complexity [SCH00, KNA00].

PuLSE-Eco

Of the 36 papers on domain analysis identified as both useful and usable by [KHU09] three of the papers directly concerned PuLSE-Eco (Economic Perspective). Even apart from those papers included in that survey PuLSE-Eco is one of the most well referenced methods in the field of domain analysis, and the most referenced method among the PuLSE methodology methods.

PuLSE-Eco is a product line and domain scoping method, focusing on asset and domain scoping but only partial product portfolio scoping [JOH06]. To decide what to include in a product line and identify potential candidate products of the product line may often be a challenging task and ha a profound impact on the future success [DEB99, SCH02b].

There are three things that make PuLSE-Eco especially interesting. Firstly as already mentioned it is a well referenced method and is compared to other methods for SPL development extensively validated in industrial case studies [SCH02b, JOH06, DEB98a, BAY99a, KNA00]. Secondly, the method has been reported to be highly customizable to the needs of the organization [JOH06]. Thirdly, PuLSE-Eco as a method is not dependent on input from the other PuLSE methods which makes it easier to use as a standalone method than some of the other methods in the PuLSE methodology.

PuLSE-CDA

The PuLSE methodologies contribution to the domain modeling area is called PuLSE-CDA (Customizable Domain Analysis)[BAY99b]. PuLSE-CDA is different from most other proposed methods as it is not as focused on the syntax and structure of the domain models but rather focuses on what to do and defines a process to do it. This makes PuLSE-CDA very flexible in its application and could even be used in combination with one or several of the below mentioned methods that does focus on for example the best syntax to use in a feature model. PuLSE-CDA is designed to take input from PuLSE-Eco and PuLSE-EM but could be used as a standalone method if the needed inputs can be provided by other means. Included in the method is also a supporting tool called DIVERSITY/CDA (Domain and Variant Engineering Supporting Technology for the Customizable Domain Analysis) that has been developed in parallel with the method. This is something that speaks well for PuLSE-CDA since good and validated tool support is rather scarce in the domain engineering field.

As with PuLSE-Eco, PuLSE-CDA has reportedly been applied in several industry projects with claimed good reuslts [DEB98a, BAY99a, KNA00], making PuLSE-CDA one of the few proposed domain engineering methods that is validated.

PuLSE-DSSA

Usually SPL development is centered around a common shared architecture. Naturally the PuLSE methodology has a method for specifying this architecture. PuLSE-DSSA (Domain-Specific Software Architecture) is the PuLSE method for creating and evaluating SPL architectures.[ANA00, BAY00a, DEB98b] The process is centered around scenarios and is reported customizable to fit small as well as large companies [KNA00].

Scenarios are created based on input from PuLSE-CDA. Relevant scenarios are selected with the help of input from PuLSE-Eco. The chosen scenarios are used to specify acceptance criteria and to design an architecture candidate. The candidate is then evaluated, redesigned and reevaluated until it meets the specified criteria [ANA00, BAY00a, DEB98b]. PuLSE-DSSA does not provide much guidance on de design of the architecture. It provides the process of deriving the design and evaluating it. This is claimed as a major advantage as it can use whatever design methodology or special architecture description that is already established in the organization [BAY99a].

PuLSE-DSSA has been applied in industry together with PuLSE-Eco and PuLSE-CDA with claimed good results [DEB98a , DEB98b, BAY00a, KNA00].

PuLSE-I

PuLSE-I (Instantiation) [BAY00b] defines a process for instantiating the generic artifacts into products. The process starts with a request for a new product. It is then characterized by iterative instantiation of the decision model and reference architecture provided by PuLSE-CDA and PuLSE-DSSA. The PuLSE-I process does not really end with the creation, testing or even delivery of the product. Instead it enters a maintenance phase where it iterates until the product becomes obsolete[BAY00b]. PuLSE-I has not as well referenced in experience reports and can not make the same claims of validity.

PuLSE-EM

PuLSE-EM (Evolution and Management) [BAY99a] defines a process for coordinating the activities of the other components. It receives work products from them and handles their evolution and maintenance. PuLSE-EM plans and guides the development of the SPL infrastructure based on the information from PuLSE-BC. Not as much has been written specifically of PuLSE-EM as of the other PuLSE methods.

11 Appendix E: Readiness Tools

It is important to know if an organization even can adopt a SPL approach or how much work that has to be done before they can, the so called readiness of the organization. Therefore there exists many tools for measuring this readiness.

Readiness Tool: Domain Potential Analysis
Authors: Sergio Bandinelli and Goiuria Sagardui Mendieta [BAN00], Goiuria Sagardui, Sergio Bandinelli and Ramón Lerchundi [SAG00]
Description: The Domain Potential Analysis describes the domain as the technical description of the domain, the market of the domain and the structure of your organization. According to this method, all these three need to be considered when identifying the domain which is one of the first steps. The Domain Potential Analysis focus on the analysis of benefits and risks then ties them together. The benefits depends on which goals the organization tries to achieve by the introduction of SPL. Some benefits will have higher priority depending on the goals. Potential benefits are, higher productivity, higher quality, higher reliability, faster time to market, better bid estimations, more on time deliveries, better life-cycle estimates, decreased cost and improved maintenance. All benefits results in the end in some kind of economic gain, direct or indirect. With this in mind they define three groups to consider for the benefits analysis, Investments, Expenses and Savings. Investments are from activities involved in the initiation of the SPL. From the maintenance of the SPL comes the expenses and finally the Savings are the difference between regular production and SPL. When analysing the risks, a Risk Analysis Model is used. This model categories the risk attributes into four factors, organization, personnel, process and products. Some attributes of the factors are the organizational structure, the experience of the personnel, if there are any SPL supporting processes and how the reuse potential of the products are. The result of this analysis are a risk profile for every attribute, factor and one for the whole domain. Then the analysis are tied together and three attitudes of organizations are used. 1) Risk Taker; Aiming for higher benefits but increasing the risks as well. 2) Risk Indifferent; Only taking more chances if there is a clear increase in benefits. 3) Risk Adverse; Aiming at reducing risks and settles for positive results. The complete analysis is visualized in a graph, with a single line illustrating the boundary for the benefit-risk ratio.
Evaluation: Quite thorough, should be suitable for organization with low domain knowledge or when changing domain.

Readiness Tool: Product Line Benefit and Risk Assessment
Authors: Klaus Schmid and Isabel John [SCH02a]
Description: This method is performed in three steps, preparation, execution and analysis. The preparation step is where the setup is performed. This includes the selection of stakeholders to interview, the definition of goal of assessment, the creation of the schedule for the assessment and the creation of the product line mapping. The actual assessment is performed in the execution step. This is mainly done by a set of structured interviews driven by questionnaires. The results from the interviews are then structured and presented to the interviewed people to receive corrections or additional information. The gathered information are in the analysis step evaluated to create a final report on the benefits and risks of adopting the SPL approach. Recommendation on which domain to choose and which risk

controlling method to use is two of the biggest results of the assessment. The final task is to initiate a meeting on site to discuss prober actions to the found results.
Evaluation: Medium effort required. May not be thorough enough to be able to make all decisions needed.

Readiness Tool: Product Line Potential Analysis
Authors: Claudia Fritsch and Ralf Hahn [FRI04]
Description: The PLPA consists of the collection of criteria, main, exclusion, inclusion and supporting. Important information that do not fall under any of these categories are placed in additional information category. Main Criteria are criteria that must exist for a beneficial SPL, the products having similar features or similar qualities and that the business develops more than one product. Inclusion criteria are indicators that some existing parts already resembles SPL parts, for example, that several products uses the same feature. If one or more of the Exclusion Criteria is being fulfilled, the SPL initiative will not generate the economic advantage that it would have otherwise. Supporting criteria are conditions that a SPL would help against, like quality or complexity problems. These criteria are checked by conducting a workshop with questionnaires. After the workshop a report is written conducting the results, rationale and some recommendations. Common recommendations are to perform a Product Line Technical Probe, to do an Architecture Trade-off Analysis Method (ATAM) [CLE02] evaluation or to perform a Scoping workshop. Two other mentioned methods are the Domain Potential Analysis and the Product Line Benefit and Risk Assessment.
Evaluation: Fast and easy method that should be good to start with.

Readiness Tool: The Technical Probe
Authors: SEI [SEI10b]
Description: This method consist of interviews of organization personnel and data analysis and uses the SEI Frameworks as reference. The Probe follows a well defined and structured process consisting of three phases, preliminary, technical probe and follow-on. First task is to put together a Probe Team which will be the main working force during the three phases. The primary goal of the first phase, the Preliminary Phase, is to gather the basic information on the organization. Including goals, status, domain, structure, terminology, process discipline and experience. Further information gathered in this phase is if there are any legacy assets, any specific SPL practices being followed and a list of available documentation. All this information provides a good overview of the organization and allows for identification of the appropriate questions to ask in the interviews, which groups to interview, what documentation to review as well as details and schedule for the second phase. In the Technical Probe phase, the Probe team starts the interviews of the designated stakeholders. The data is collected with care and with strict confidentiality to preserve and not alter any data. The data is then analyzed in context of the practice areas of the SEI Framework. The data is then characterized as observations, strengths, challenges and recommendations for each practice areas as well as overall for the whole organization. In the final phase, the follow-on, the Probe team puts together the report with the findings of the Probe. This should include the Action Plan with description of how to handle the found challenges and how to use the identified strengths. Also there should be descriptions on how to execute the action plan, specially in areas involving the following areas, architecture definition, asset extracting, scoping, commonality analysis as well as the development of business case, operating concept, SPL metrics and acquisition strategy.
Evaluation: Thorough method that is tightly based on the SEI Framework which makes this a desired approach if there is available time and money for it.

Readiness Tool: Knowledge Assessment
Authors: Stan Bühne, Gary Chastek, Timo Käkölä, Peter Knauber, Linda Northrop and Steffen Thiel [BÜH04]
Description: According to Büher et al. readiness is measured by collecting knowledge which is gained by answering certain questions. These questions handle four different areas, the Market, the Organization, the Business Unit and the Individuals. Question from the market area aim to answer how big and mature the market is and which if any that are the market leaders. Other questions to answer is if there are any open standards or safety criteria for the products in the market. When investigating the organization, you must know what the current position of the organization is, how good contact the organization has with its customers and how much control the organization has over its product specifications. Other things to find out are, how well the management are committed, how fell funded the SPL initiative are and how motivated the personnel of the organization are. When it comes to the Business Unit, the mayor questions concern how good the business units are, experience, current process discipline, domain-specific knowledge and software engineering especially architecturing. Further questions to ask concerning the Business Units are if there can be assets extracted from legacy products, how much the current workflow differ from SPL workflow and if there is a clear business strategy. And perhaps the most extensive and most important question to answer, can the business unit handle the 31 practice areas of the SEI framework. Questions concerning the Individuals is asking if they have the necessary capacity to learn SPL and the necessary expertise to perform the SPL tasks, and if not, how they will get it. How motivated they are along with if the individual gains or loses on the SPL introduction are two more question that should be answered.
Evaluation: A bit different from the rest of the methods, based on the SEI Framework. Should give the organization good insight on the readiness of certain areas and thus information where they need more work.

12 Appendix F: Adoption Approaches

Suggested by Researchers

Type: A knowledge based transition approach

Source: Gerardo Matturro and Andrés Silva in [MAT05]

Description: Proposes that the transition to SPL should be faced as a Knowledge Management problem. Introducing SPL leads to change in the organization's business strategy which they mean can lead to huge knowledge gaps between the knowledge the organization has and the knowledge it must have. To deal with this they present a method for identifying and assessing these knowledge gaps. First they define three types of knowledge, general, particular and specific. They choose to focus their assessment on two elements, working experience and formal training and study. Then they start assessing what knowledge the organization needs to successfully establish a SPL. This is done with help of a template that extracts the required level of the two elements on every PA of the SEI Framework. Next step is to establish what the organization knows. This is done by constructing general personal profile and practice areas organizational profile. Final step is to identify the gaps, but this step is not well described.

Evaluation: Knowledge is important, no doubt, but this approach might not be executable. It is a bit brief and not validated.

Type: An adopting and institutionalizing approach

Source: Günter Böckle, Jesús Bermejo Muñoz, Peter Knauber, Charles Krueger, Julio Cesar Sampaio do Prado Leite, Frank van der Linden, Linda Northrop, Michael Stark and David Weiss in [BOC02].

Description: Describes how to perform the transition to product line engineering and lists the various strategies for such a transition. It also describes how to create an adoption plan and how to institutionalize product line engineering in an organization. Describes the overall picture then in smaller steps, starting with goals and stakeholders and then business cases which shows how just the different stakeholders goals are accomplished by the SPL. Third step is the Adoption Plan, which quite obvious describes how to perform the transition. The adoption plan is made of three things: 1) A characterization of current state: This is done by characterizing many business parameters, including the process, the staff, the organizational structure, the management methods and the engineering methods. 2) A characterization of desired state: This is done by characterizing the same parameters for the desired state as for the current state for easy comparison. 3) Strategies, objectives and activities: Basically the things needed to get from the current to the desired state. This includes what approach to use, they describe three, Incremental, Pilot and Big Bang. If plan is accepted the SPL is launched. Some activities involved in the launch is stakeholder identification, determination of roles and responsibilities, definition of SPL process, definition of production process, planning of funding and staffing and finally some initial data collection. Final step of the adoption process is to institutionalize the SPL so that the involved managers and staff will consider it as part of their working culture. A SPL is institutionalized when the product line processes are considered stable and indispensable to that organization.

Evaluation: Very thorough and written by many big names like Linden and Northrup. Very general and should work for most organizations, but this can be a drawback as well, since then its not specifically created for a certain type of organization. No validation.

Type: BigLever Software's transition approach
Source: Charles Krueger in [KRU02a]
Description: This transition approach were created to lower the adoption barrier, meaning the upfront costs, the level of effort, the assumed risks and the latency required. The approach supports three transitions models, reactive, proactive and extractive. Reactive is acting on demand, proactive is acting in advance and extractive is building the SPL based on existing products. It uses separation of concerns to simplify SPL which is the process of separating the systems into distinct features. A concern could be any piece of interest or focus in a certain system and are typically the same as feature. These features should have as little of the same functionality as possible, hence separation of concerns. The three models are quite ordinary and not that novel. It is concluded with a six step example.
Evaluation: Well illustrated with graphs and a thorough example. Might not be that novel except for the extractive approach. This approach is according to us not a approach in itself, instead it is a method of acquiring assets.

Type: An evolutionary lightweight iterative adoption process
Source: Daniel Simon and Thomas Eisenbarth in [SIM02]
Description: This lightweight iterative process supports the incremental transition towards a SPL approach. First they present the prerequisites which are almost none, only that the system and software are understood by the programmers and users. The participants of this approach are an expert programmer, an architect of the legacy system, a domain engineer, some users and customers, a reengineer and a product line expert. The process itself consists of several steps: 1) Identification of available and accessible features. 2) Prioritizing of identified features. 3) Market analysis to anticipate future features. 4) Reengineering with help of Feature analysis. Feature analysis consists of four substeps: 4.1) Locating where the features were implemented in the legacy code. 4.2) Recognizing and locating the commonalities and variabilities of the legacy systems. 4.3) Recognizing the dependencies among the features. 4.4) Evaluating if the intended architecture is feasible for a SPL approach why using metrics. 5) Effort estimation and creation of PLA based on the feature analysis. 6) Reengineering of the assets according to the effort estimation and the PLA. These six steps could then be iterated and during and after each iteration the following artifacts are updated, the PLA, the product feature list, the feature map and the software sources.
Evaluation: Well described, novel and well referenced approach with an easy to follow validation of method with an example.

Type: Adoption Factory Pattern
Source: Paul Clements, Lawrence Jones, John McGregor and Linda Northrop in [CLE06].
Description: This roadmap is in form of a adoption factory pattern which gives the organization the answer to what they need to focus on next. It is built out of a three times three grid where one side is what part of the SPL and the other is what time

in the adoption process that the organization currently is. The three parts is process product and organization, and the three states are establishing context or “cold start”, establishing production capability or “in motion” and operate production line or “monitoring”. Besides this they include which of the SEI frameworks 29 practice areas that belong to which part and state. They conclude with an example with an hypothetical company.

Evaluation: An easy to follow, well written, short and consist roadmap for adopting SPL. Somewhat validated and should be plausible.

Type: A revolutionary initiation approach

Source: Mari Matinlassi, Eila Niemelä and Liliana Dobrica in [MAT02].

Description: QADA is as the name suggest a method for creation and evaluation of system architectures based on quality attributes such as maintainability, reusability, modifiability, adaptability and portability. It also includes a requirement engineering phase that start the method. Included in both the requirement engineering and in the analysis are domain analysis activities. QADA considers architecture on two abstraction levels, conceptual and concrete. On both of these abstraction levels there is also produced architectural descriptions on three viewpoints, structural, behavioral and deployment. These differ between conceptual and concrete levels. In the conceptual level the structural viewpoint refers to the composition of software components, the functionality that the product must have. The commonalities and variabilities are identified in the structural view and some quality attributes such as adaptability, reusability and portability are considered. The behavioral viewpoint takes behavioral architecture aspects under consideration to be able to understand quality requirements such as reliability, performance and security. These three quality attributes are considered during the deployment view also along with availability, capacity and bandwidth cause the deployment view concerns the embedding and allocation of software components. In the concrete level the structural view is concerning matters like refining components and interfaces between them. The behavioral viewpoint states the behavior of each separate component and creates state diagrams and sequence charts. In the last view of the concrete level, concrete executable software components are created. The quality of the architecture on both abstraction levels on all three views are then analyzed in the corresponding analysis phase by addressing stakeholders, architecture and design in the conceptual level and by extensive scenarios in the concrete level. Overall it is a fairly traditional architecture design method with the addition of and focus on configuration and customization concepts that are vital for PLA. Included is also some guidance on how to use the approach as well as internal case study with successful results.

Evaluation: As it is fairly similar to common methods for single system architecture design. QADA should not be hard to adopt by a company that are used to follow such an architecture design method. That requirement engineering, domain analysis and architecture evaluation is included in the method makes the method more usable compared to a method that relies on other methods to provide input. There are a lot of superfluous text but at the same time it is well written and has very thorough explanations with illustrations to clarify. The case study on the approach is somewhat diffuse, but should give the approach some validation.

Type: Enhanced Unified Process for Product Line (UPEPL)

Source: Weishan Zhang and Thomas Kunz [ZHA06]

Description: This model is based on the Unified Process which facilitates reuse for a single system, but lacks when it comes to handling multiple products. Because of this the Enhanced Unified Process for Product Lines was created. Product line activities were added and put together with regular UP activities, thus taking advantage of both Unified Process and software product line practices. The main characteristics of regular UP is, using iterative and incremental development, centering around software architecture and embracing change by considering feedback. UP consists of four phases and the first phase, the Inception phase, is about considering if a SPL is profitable and doable. In the Elaboration phase which is the second phase, the architecture and thus the foundation is created and validated. The development of the systems are the main task of the Construction phase. In the fourth and final phase, the Transition phase, the product is delivered to the user. The SPL practices added in UPEPL is in the first phase SPL scoping, domain analysis and initial feature model creation. In the Elaboration phase a SPL architecture is defined and refined, meta components for the core assets are created and tested based on the requirement plan which itself are refined. As the whole SPL is refined the final feature model is created base on the initial. In the construction phase, the last meta components are created and tested and some of them are built into new products as the product line is ready. Unit testing and integration testing are also performed in this phase. The product is evaluated against the acceptance criteria in order to make a smooth transition to the end user in the transition phase.

Evaluation: A well referenced and well written adoption process that is based on the well used UP model.

Type: Staged adoption of software product families

Source: Jan Bosch in [BOS05].

Description: Starts with describing the most predominant problems and what to do to avoid them. Then presents five decision dimensions representing the most important decision that an organization have to take. First dimension is feature selection which is about selecting which features that should be moved first from regular to SPL development. Secondly, there is the architecture harmonization, this one is handling the creation of the architecture so that it fits with the products. R&D organization, the third decision dimension, is concerning the research and development teams that are to develop the shared components. Funding model is the fourth dimension and is about how to get the funding needed. The fifth and final decision dimension is shared component scoping, and is basically how to develop the components. For all of these decision dimensions there exists some different approaches, these approaches are described with advantages and disadvantages. After this, he describes the three phases of initiating the SPL and for each decision dimension he described what decisions that are the preferred ones for each of the three phases. The three phases are, initial adoption, expanding scope and increasing maturity.

Evaluation: Good adoption approach, the decision dimensions seems like a good way to tackle the transition process. Well written and well referenced.

Used in Industry

In this appendix we discuss the approaches tried and used in industry. The findings are structured with name of article containing the case study and a short description

and evaluation of the approach used. Those case studies that did not describe the process used are not presented here though they were of no value to us.

Case Study: "A Case Study in Applying a Product Line Approach for Car Periphery Supervision Systems"

Source: Steffen Thiel, Stefan Ferber, Thomas Fischer, Andreas Hein and Michael Schlick [THI01]

Description: They divided their adoption process into four parts and started with scoping. Scoping was about defining the boundaries of the domain, the core and the borderline functional areas, the constraints as well as the business, the organizational, the technical and the legal requirements of the SPL. Requirement analysis and modeling, was the second step and it started with a domain analysis phase. They gathered, described and classified the requirements for the different product variants and stated whether each requirement was functional, qualitative or if it was a design constraint. Third step was feature modeling, this was made up of a feature tree, composition rules and rationales. A feature tree is a tree structure with the features interrelated by mandatory, alternative and optional branches. Composition rules define allowed combinations of optional and alternative features and rationales express concerns to consider when selecting the features. The fourth and final part was architecting and design, here the three earlier parts laid the ground for the architecture. The architecture consisted of the technical structure of the system with both software and hardware components and connectors and also compositions of them.

Evaluation: A good case study on an adoption of SPL in a system. Detailed descriptions of reason for SPL. Only one iteration made and explained but the approach was constructed to support multiple iterations to perfect the outcome. Well referenced and well written.

Case Study: "Building software product line from the legacy systems experience in the digital audio & video domain"

Source: Kangtae Kim, Hyungrok Kim and Woomok Kim [KIM07a]

Description: This approach is built out of four steps with a total of 18 substeps. Besides these substeps which are the main activities, there are some additional activities, reengineering activities as well as tools to use. Each step also states what outputs or deliverables that step leads to. First step is feature identification and analysis and the primary objective of this step is to analyze the platform requirements based on results of legacy system analysis and the requirements of the platform itself. To achieve this goal it has the following main activities, identify and analyze features, feature modeling and elicit requirements. Besides these, this step has activities such as analyze of legacy systems, legacy architecture and the technical roadmap, sketching of candidate architecture and adoption of corporate requirement standards. Second step is the platform analysis and activities belonging to this step is defining of architectural requirements, analyze of static and dynamic model and design and evaluation of candidate architecture. Additional activities here are benchmarking the platform case study, feature analysis with legacy systems, applying corporate platform design principles and applying ATAM as evaluation tool. Aim of the second step is to design some candidate architectures and to identify core assets of the future SPL by both static and dynamic analysis. Third step is platform design and evaluation and now its time to finalize the architecture and make it the reference architecture. Defining platform configuration policy and interface standards, design and evaluate the

architecture and design components are the main activities of this step. Additional activities are evaluation of platform configuration methods, defining of UML modeling rules and disciplines, analyze of legacy systems interfaces and evaluate architecture by corporate platform design principles and ATAM. Last step it to build and test the platform by refining of development environment and then implementing, unit testing, system testing and finally deploying/releasing of platform.

Evaluation: A thorough and well written adoption approach, including tools to use for the activities as well as clarifies the deliverables you should get. Also includes guidelines on how to perform the activities especially how to create the architecture.

Case Study: "Conditions and restrictions for product line generation migration"

Source: Mikael Svahnberg and Michael Mattsson [SVA02]

Description: This adoption process is divided into three steps, the steps are create architecture, evaluate the architecture and develop the migration plan. First step starts with a workshop with the company team and an external architecture team. Preparations for the workshop for the architecture team is to read literature and manuals for existing system. The company team prepares themselves by reading up on architecting, creating list of requirements and creating a collection of scenarios. The workshop itself starts with making the requirements and scenarios clear, unambiguous and understood by all involved. Next step of workshop is to iteratively design the architecture with help of integration of scenarios. This architecture created is a only functionality based architecture. After this the market department supplies market scenarios that the architecture must handle to verify that the architecture fits the market view of the organization. Additional revising based on evaluations, remarks and discussion is conducted then. This created architecture is used to convince management to proceed with this process. Then the architecture is revised to incorporate the functionality and quality attributes required. Second step is to evaluate the architecture, and this is done by the method ATAM. Third and final step is to create the transition plan, and included in this are decisions on how existing component and subsystem should be migrated to or excluded from the newly created architecture.

Evaluation: Good article, well referenced and well written. Includes challenges and foreseen problems. A bit brief descriptions.

Case Study: "Experiences with product line development of multi-discipline analysis software at Overwatch Textron systems"

Source: Paul Jensen [JEN07]

Description: This adoption approach consists of four phases, preliminary steps, starting out, course correction and current state. Main activities in the first phase is scoping, market analysis, performing a technical probe and training of staff on product line concepts. Additional activities are the development of fundings model and organization model and the forming of a domain engineering group. In the second phase, starting out, the initial architecture was developed focusing mainly of dependencies between components in the SPL. Also performed in this phase were the mining of asset from legacy code. In the third phase a new architecture and infrastructure was created. Final phase is the integration and launching of products. Also provides a summary of the lessons they learned during the transition process.

Evaluation: Lacks in description on how they performed each step, focuses on characteristics.

Case Study “HomeAway's transition to software product line practice Engineering and business results in 60 days”

Source: Charles W. Krueger, Dale Churchett and Ross Buhrdorf [KRU08]

Description: They use the 3-Tiered SPL Methodology which consists of three key elements. The first element is the base tier with advanced variation management and fully automated production that results in optimized developer productivity and massive reduction of costs. Middle Tier is the next element and now the development shifts from product focus till asset focused. There is also an increase of reuse and asset expertise which leads to optimized product quality. Last is the top tier where the business portfolio moves from product based to feature based. The portfolio evolves by adding or refining feature requirements which results in extremely efficient collaboration between business and engineering.

HomeAway have based their SPL transition on this model and have created specific objectives for each tier. The first tier has five objectives and the first one is to remove the one size fits all approach. Secondly, enable independent and separate development for each of the different sites. Next objective is to create a smaller runtime footprint for each development site. Fourth objective is to more efficiently create, evolve, maintain and manage feature variant among different sites. Last objective is to increase test coverage and eliminate redundant regression testing on all sites when a site changes. The Middle Tier has four objectives and first is to modularize the monolith. Second is to incrementally refactor and reengineer legacy system as SPL assets in an effective way. The two last objectives are to utilize the provided variation points and establish better organization around asset teams. The Top Tier has in Home Away four objectives. The first two of these are to have faster roll out of features and to have better configuration of features and diversity. The two other goals are to improve communication between marketing and engineering and to have a optimized roll out of new sites.

Evaluation: This approach is very specific for this case and goes into code detail. It lacks general descriptions on the methods used and focuses on the results and not way the work are performed.

Case Study: “Incremental return on incremental investment Engenio's transition to software product line practice”

Source: William A. Hetrick, Charles W. Krueger and Joseph G. Moore [HET06].

Description: This transition process consists of four primary stages, each of these has incremental structure and is made by small incremental investments to disturb the regular development as little as possible. First step is transition the infrastructure and core assets and this step included activities such as performing a pilot project, create the SPL development infrastructure, extract a core asset base from legacy products, incrementally move the teams to SPL development and refactor the assets to improve commonalities and variabilities. Transition the team organization from product based to core asset based is the second step. Here each team gets a core asset manager and a core asset technical leader. The core asset manager is responsible for making sure that the core assets work as intended. The core asset technical leader has the responsibility to make sure that the core assets gets implemented according to the architect and feature requirements specified by the product roadmap. They both together have an additional responsibility of maintaining integrity of the core assets. Main activity in the second phase is to define the asset teams and to train team members, leaders and

managers in their roles and responsibilities. Next phase is the transition of the development processes from being product released based towards feature released based. The major investment of this phase was to assign a task force with the purpose of creating a SPL process that satisfied the need of the company. The company needed to increase effectivity of communication, increase compatibility with SPL practices, include an asset assembly function, include a product validation function and include a mapping step on how to move from feature requirement to asset requirements. Final step is the transition of the validation and quality assurance from individual products to SPL assets. This step includes changers such as completing all the feature validation iteratively as part of the development and shifting some focus from product certification to interoperability management.

Evaluation: Good case study. Stepwise introduction that gave ROI after each step, and thus building up confidence. This approach is complicated and a bit hard to follow but at the same time it seems good. The transition process was when this case was written not complete but the first and most important steps were conducted.

Case Study: "Introducing PLA at Bosch Gasoline Systems experiences and practices"

Source: Mirjam Steger, Christian Tischer, Birgit Boss, Andreas Müller, Oliver Pertler, Wolfgang Stolz and Stefan Ferber [STE04]

Description: Experience report from introducing SPL. Lot of consequence discussions and their solutions. They introduce the SPL at the same time as they are doing CMMI stuff such as process improvement, advancing in process maturity.

Their process start with market analysis and scoping by considering the market, the current platform and the business strategy. This market analysis defines goals and qualities for when to define the PLA and for the evolutionary redesign of the existing assets. After the market analysis they investigated and customized the PLA. To find strengths and weaknesses they applied the PLTP tool. Based on this, new methods and processes were developed and some existing were modified to address these weaknesses. Those were then rolled out and institutionalized. Along with this the existing assets were redesigned. After this the new platform were design and implemented with help of feature analysis. Final step of the adoption process were to perform requirement analysis and product design, implementation and finally integration.

Evaluation: A bit unclear since they focus on consequences and solutions and not their specific practices. The process steps are because of this hard to follow.

Case Study: "Introducing product lines in small embedded systems"

Source: Christoph Stoermer and Markus Roeddiger [STO02]

Description: This approach consists of six transition phases. Activities of the first phase, Investigation, are reconstruction of the existing products and also elicitation of the requirements. In the Architecture Design phase, Attribute Driven Design created by SEI are used to design the product line architecture. The third phase is the Demonstrator phase, and in this phase the verification is designed. Production is the fourth phase and now the main focus is the get a complete product cycle involving development, testing, maintenance and customer contacts. In the fifth phase, the Organization phase, separation of core asset and application development is the main objective. It is done by defining interfaces and integration. Domain is the final phase and here domain engineering are introduced. This approach emphasizes that the

organization takes the transition in its own pace. This is to reach a high level of commitment and to build the necessary foundation. And importance are put on personal skills like integrity and communication.

Evaluation: Only three phases thoroughly explained since the company have not come further. The adoption approach seems promising though.

Case Study: "Observations from the Recovery of a Software Product Family"

Source: Patricia Lago and Hans van Vliet [04]

Description: Well constructed case study on an organization creating a SPL from six previously developed products. They do this by reverse engineering of the six products to define commonalities in features and components as well as traceability among them. They started by characterizing the problem space by defining a classification of the domain features. To get a better overview of the commonalities they defined three different layers, the application layer, the middleware layer and the platform layer and then investigated for each layer which components were included in the six products. After this they started the recovery of the features. This was done by iterative procedures involving studying, reverse engineering and redefining of the documentation, the design and the code itself which in turn was done in five subsequent steps. After this a new product was introduced in three steps which led too two steps of updating the product family feature map and product feature maps. For more detailed information on the recovery of features refer to [LAG02].

Evaluation: A case study focusing on reengineering legacy products with clear steps and activities. Well written and well referenced.

Case Study: "The JTRS program Software-defined radios as a software product line"

Source: Eric Koski and Charles Linn [KOS06]

Description: Military funded case study doing revolutionary approach with specific core asset development units. They create their framework architecture using CORBA and component based design. They do this with a thorough understanding of their domain and a carefully designed strategy for acquisition, testing and validation. The architecture is defined by a framework architecture to simplify the process.

Evaluation: This approach is not plausible for most organizations cause it requires massive funding, which in this case is provided by the US military. Lacks a bit on how they perform their tasks.

Case Study: "Transitioning to a software product family approach - challenges and best practices"

Source: Michael Kircher, Christa Schwanninger and Iris Groher [KIR06]

Description: Part case study, part survey, states a set of practices in form of what to do and what to avoid, which has worked for them. First is to have a software architect that understand the importance of scalability. The architect must 1) Communicate the requirements and design guidelines for the rest of the organization. 2) Make sure that there is consistency in the architecture by reviewing all design made by the individual designers. 3) Guide the developers on how to create good design. 4) Contribute design knowledge. Next, the customer requirements should be separated from the technical requirements, which eases the creation of the design. Design should become more clear, consistent and concise. Next, the organization must have a certain maturity before initiating the SPL approach. Another important thing that the

organization should have is some sort of mechanism or implementing and monitoring variability. This can be done in a number of ways which all falls within two different categories, another level of indirections and language and generative support. Last but not least is to have close collaboration between management and development, they should not only exchange specifications. Management should not only view the product as a black box but should lead and monitor the whole development and especially the core asset development. These practices are validated and tested properly, and besides these they also have some practices that isn't as validated but still seems to work. First they follow a set of guidelines in the introduction, assessment, awareness, scoping, first seed, involve development, shape organization and sharpen the saw. When it comes to domain modeling and variant management, they follow a few simple steps 1) Identification of terminology and constraints. 2) Problem space description. 3) Variability analysis. 4) Solution space description. 5) Mapping of problem space to solution space. 6) Tracking. Also they have some guidelines on how to set up the responsibilities for variability management. The whole organization must participate and it should be performed by every staff involved and then reviewed and coordinated by one single person.

Evaluation: A good article which contains much useful information. It is well referenced and well written. Lacks in execution of steps.

Table 11: Adoption Approaches Comparison

Adoption Approach	Source	Derives from	Validation	Repeatability	Novel	Adoption Type
A knowledge based transition approach	[MAT05]	Research	None	Medium Low	Yes	Evolutionary
A revolutionary initiation approach	[MAT02]	Research	Hypothetical Example	High	Yes	Revolutionary
Adoption Factory Pattern	[CLE06]	Research	Hypothetical Example	Medium High	Yes	Several Types
An adopting and institutionalizing approach	[BOC02]	Research	None	Medium High	Yes	Several Types
An evolutionary lightweight iterative adoption process	[SIM02]	Research	Hypothetical Example	Medium High	Yes	Evolutionary
BigLever's Software's transition approach	[KRU02a]	Research	Hypothetical Example	Medium High	No	Several Types
Case Study at Bosch Gasoline Systems	[STE04]	Industry	Industry Case Study	Medium Low	No	Evolutionary
Case Study at Engenio	[HET06]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study at HomeAway	[KRU08]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study at Overwatch Textron Systems	[JEN07]	Industry	Industry Case Study	Medium	No	Evolutionary
Case Study at Siemens AB	[KIR06]	Industry	Industry Case Study	Medium	No	Evolutionary
Case Study in Car Periphery Supervision Systems	[THI01]	Industry	Industry Case Study	Medium High	No	Evolutionary
Case Study in Small Embedded Systems	[STO02]	Industry	Industry Case Study	Medium Low	No	Evolutionary
Case Study in the Digital Audio & Video Domain	[KIM07a]	Industry	Industry Case Study	High	No	Evolutionary
Case Study on systems for automatic guided vehicles	[SVA02]	Industry	Industry Case Study	Medium High	No	Evolutionary
Enhanced Unified Process for Product Line	[ZHA06]	Research	Industry Case Study	Medium High	Yes	Evolutionary
Military Case Study the JTRS program	[KOS06]	Industry	Industry Case Study	Medium Low	No	Revolutionary
Observations from Multiple Case Studies	[PAT04]	Industry	Industry Case Study	Medium	No	Evolutionary
Staged adoption of software product families	[BOS05]	Research	Low	Medium	Yes	Evolutionary

Derives from: States whether the approach comes from researchers or from industry.

Validation: States the level of validation of approach. Low are when the approach is based on previous experience, hypothetical example is small internal made up case

study and industry case study are when the approach has been used in industry and thus gained validation.

Repeatability: Defines how good the approach can be repeated and used again. Thorough descriptions, illustrations and examples increases the repeatability.

Novel: This is a yes or no question on how novel the approach is, novel is not always a good thing.

Adoption type: Simply states what adoption type it belongs to. Can basically be evolutionary or revolutionary, some defines or can be used for multiple types though.

13 Appendix G: CMMI and SPL Summary

CMMI Background

Included in CMMI are five definition on how mature your processes are, they are Performed, Managed, Defined, Quantitatively Managed and Optimized. A Performed Process is a process that just does what it have to and nothing more. A Managed Process is a Performed Process but it is planned and executed with regards to policy. In addition a Managed Process is performed by staff that is skilled and have sufficient resources to produce the outputs. A Defined Process is in addition to the attributes of a Managed Process based on a set of standard processes and with help of guidelines on how to create new processes. It also have a maintained process description and assets from the process like work products and measures are added to a process asset library. Another improvement in Defined Process against Managed Process is that the Defined Processes are general and not specific to a single project like a Managed Process. Finally a defined process is more detailed and the management of a Defined Process is based on an understanding of the interrelationships of the activities, measures, work products and services of the process. Next grade of maturity of processes is called Quantitatively Managed, which is a Defined Process that is statistically controlled with quantitative techniques. The quality of processes and services are measured and controlled through the whole project. Quantitative objectives and activities are established, these are based on capability of the organization and its employees. Using these quantitative methods increases the predictability of processes significantly. The highest grade of process maturity is the Optimizing Process which is a Quantitatively Managed Process that is adapted to meet the current business objectives. It is also optimizing process performance by incremental and innovative improvement. Improvement that address issues with defects, process variations and other problems.

To better suite all kinds of organizations the CMMI team has constructed an additional representation that differs from regular or staged representation. This new way of working with CMMI is called continuous representation. Having Continuous Representation means that you focus on the specific processes that is most important for your organization. These processes might be needed to achieve goals or they might be of high risk. The Staged Representation on the contrary is designed with a standard sequence of performing the improvements and thus serve as a basis for evaluating the maturity of the organization. Continuous Representation offers maximum flexibility for using a CMMI model while Staged Representation offers a systematic and structured way of performing process improvement one stage at a time. Continuous Representation is a good choice if you know which processes that need improvements and have a understanding of your processes and their dependencies. If your more uncertain on how to start and not have full knowledge on your processes then Staged Representation is the way to go since this representation has been determined by more then a decade of research and improvement cases.

Which representation that is preferred depends on the organization. But since there are certain process areas that are more important for SPL development, continuous might be the better choice. This because you can choose which Process Areas to perform and at which rates. For example, an organization with maturity level three might want Causal Analysis and Resolution which is a level five process area and

extra important for SPL development. A drawback with Continuous Representation though, is that it is a newer approach and do not have as much data from cases to support and guide the process. Also Staged Representation gives a predefined and proven improvement path, but for single product development and this path might not work for SPL development. If the set of Process Areas that belong to each maturity level could be altered a bit and modified to suite SPL better, then Staged Representation should be the best choice.

Beneficial Process Areas

We here present the complete list of Process Areas in CMMI, descriptions are extracted from SEI homepage on CMMI [SEI10a] and belonging publications on CMMI available on said homepage. The evaluations attached to the PA's are made by discussions between us. Final comments are a short total grade on the PA's importance in SPL. Grades consists of: (++) Crucial, (+) Important, (0) Equally important, (-) Less important, (--) Unimportant, (*) Important in certain cases, (!) Extra difficult.

Table 10: Definitions of importance

(--) Unimportant	This PA lack relevance to SPL development.
(-) Less important	This PA is not as well utilized in SPL development or there are other SPL related processes that work around this PA.
(0) Equally important	The baseline. We could not find any differences with this PA when having SPL development processes compared to traditional development.
(+) Important	SPL development practices utilize this PA a lot. Putting extra effort at this PA may increase the effectiveness of the SPL.
(++) Crucial	SPL development practices rely on this PA. Poor execution of this PA may cause the SPL to fail.
(*) Important in certain cases	This PAs importance depends on what SPL development methods and techniques are used. Otherwise it is equal to (+).
(!) Extra difficult	When using SPL development you will have to take this PA one step further regarding analysis or execution.

CMMI Maturity Level 1 – Initial – Process Areas

No process areas on this maturity level.

CMMI Maturity Level 2 – Repeatable – Process Areas

Process Area: Configuration Management (CM)

Purpose: To establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting and configuration audits.

SPL Comments: This PA is extra important for SPL because the reusable assets evolve and change as they are used in new products. After some time there will be many versions of the assets in circulation and in different configurations of the products. Keeping track of these version and releases is important for the maintenance process. When the number of products increases and the SPL complexity grows this PA gets more complicated.

Final Grade: ++!

Process Area: Measurement and Analysis (MA)

Purpose: To develop and sustain a measurement capability that is used to support management information needs.

SPL Comments: To base managerial decisions on accurate measurements of current processes and results are vital for any engineering discipline. This is no different for SPL engineering.

Final Grade: 0

Process Area: Project Monitoring and Control (PMC)

Purpose: To provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.

SPL Comments: This process area is no different for SPL engineering. The projects may have a different shape but they still need to be monitored and controlled.

Final Grade: 0

Process Area: Project Planning (PP)

Purpose: To establish and maintain plans that define project activities.

SPL Comments: The process area as such is no different for SPL's, but the plans and activities will be different from single system development. Extra importance should be put on this process area if an evolutionary approach is chosen since the asset development plan will extend over many product development projects.

Final Grade: *

Process Area: Process and Product Quality Assurance (PPQA)

Purpose: To provide staff and management with objective insight into processes and associated work products.

SPL Comments: That the processes and produced artefacts are understood are very important in SPL engineering. This prevents the likelihood that shortcuts are taken which in turn prevents corrosion of the SPL.

Final Grade: ++

Process Area: Supplier Agreement Management (SAM)

Purpose: To manage the acquisition of products from suppliers.

SPL Comments: Reusable assets can be acquired from suppliers either by commission or by buying COTS. In these cases this process area gains importance.

Final Grade: *

Process Area: Requirements Management (REQM)

Purpose: To manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.

SPL Comments: The requirement management is more complicated for SPL since the requirement originates from multiple products. Not only can there be inconsistencies within the requirements and work products of one product but it is also a risk that such inconsistencies will arise between products that are to share assets. Inconsistencies within one product can not be corrected without analysing the consequences for all the other products. This makes this process area extra important in SPL engineering.

Final Grade: +!

CMMI Maturity Level 3 – Defined – Process Areas

Process Area: Decision Analysis and Resolution (DAR)

Purpose: To analyse possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria.

SPL Comments: This is a very important process area in SPL engineering. As new products are introduced and requirements change for old products the SPL will have to evolve. Each change request and its consequences on all products in the SPL will have to be analysed. What changes to include in the SPL and what changes should be made product specific severely effects the complexity, reusability and configuration management of the SPL. A different SPL area where the DAR process area would be highly involved is the Make, Buy, Mine, Commission analysis made when developing new assets.

Final Grade: ++

Process Area: Integrated Project Management (IPM)

Purpose: To establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard processes.

SPL Comments: In SPL asset development there will likely be a larger number of stakeholder since it in involves several products and possibly a larger span of customers and end users. This can make this process area more complicated but also more needed.

Final Grade: +!

Process Area: Organizational Process Definition (OPD)

Purpose: To establish and maintain a usable set of organizational process assets and work environment standards. For IPPD, Organizational Process Definition +IPPD also covers the establishment of organizational rules and guidelines that enable conducting work using integrated teams.

SPL Comments: This is of course an important process area although we can not think of any SPL specific situation when this would more or less important than for single system development.

Final Grade: 0

Process Area: Organizational Process Focus (OPF)

Purpose: To plan, implement, and deploy organizational process improvements based on a thorough understanding of the current strengths and weaknesses of the organization's processes and process assets.

SPL Comments: This is of course an important process area although we can not think of any SPL specific situation when this would more or less important than for single system development.

Final Grade: 0

Process Area: Organizational Training (OT)

Purpose: To develop the skills and knowledge of people so they can perform their roles effectively and efficiently.

SPL Comments: As SPL development differs significantly from single system development on several points there will probably be a increased need for this process area. An organization that uses SPL methods and techniques can not presume that new employees will have the skills and knowledge needed. This process area will also play a large role in creating acceptance and support for a new SPL initiative.

Final Grade: +

Process Area: Product Integration (PI)

Purpose: To assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product.

SPL Comments: This process area is actually a good description of product production in a SPL. As component (asset) integration is a large part of SPL development this process area can be considered of special interest to SPL engineering.

Final Grade: ++

Process Area: Requirements Development (RD)

Purpose: To produce and analyze customer, product, and product component requirements.

SPL Comments: It is important for SPL engineering to elicit the correct requirement for all products, and also to establish which requirements are mandatory, optional or variants. This process area will likely be more complicated for SPL engineering as there are several products and the reusability to take into consideration.

Final Grade: +!

Process Area: Risk Management (RSKM)

Purpose: To identify potential problems before they occur so that risk-handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives.

SPL Comments: The risks in SPL development may be a bit different at times but the risk management process area is unchanged.

Final Grade: 0

Process Area: Technical Solution (TS)

Purpose: To design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related lifecycle processes either singly or in combination as appropriate.

SPL Comments: A note on technical solutions in SPL's is that they should be independent of the products. There may also be technical solutions on several abstraction levels where the higher abstraction is product independent and the lower abstraction are product dependent variation of the higher abstraction. The use of SPL in itself is also a technical solution. But we ca not think of any SPL specific situation were this PA is more important.

Final Grade: 0

Process Area: Validation (VAL)

Purpose: To demonstrate that a product or product component fulfills its intended use when placed in its intended environment.

SPL Comments: This is very important independent on how the product and components are developed. For SPL engineering there are different strategies to perform this that are either very complex or sacrifice some of the reusability concept of SPL. This has to do with the fact that to ensure that the components fulfills their intended use they will have to be tested in every configuration imaginable which is practically impossible. See Quality Assurance and Testing in section 4.2.6.

Final Grade: ++!

Process Area: Verification (VER)

Purpose: To ensure that selected work products meet their specified requirements.

SPL Comments: The requirements on SPL assets can be quite extensive as they are to be used in many products and in many configurations. This makes this process area harder to execute.

Final Grade: !

CMMI Maturity Level 4 – Managed – Process Areas

Process Area: Organizational Process Performance (OPP)

Purpose: To establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and to provide the process-performance data, baselines, and models to quantitatively manage the organization's projects.

SPL Comments: This is of course an important process area although we can not think of any SPL specific situation when this would more or less important than for single system development.

Final Grade: 0

Process Area: Quantitative Project Management (QPM)

Purpose: To quantitatively manage the project's defined process to achieve the project's established quality and process-performance objectives.

SPL Comments: This is of course an important process area although we can not think of any SPL specific situation when this would more or less important than for single system development.

Final Grade: 0

CMMI Maturity Level 5 – Optimized – Process Areas

Process Area: Causal Analysis and Resolution (CAR)

Purpose: To identify causes of defects and other problems and take action to prevent them from occurring in the future.

SPL Comments: Should be applied early in SPL development. Finding the cause of defects and correcting them in the right assets prevents the defects from being introduced in future products while maintaining the SPL integrity. Finding out why the asset was defect and correct the processes will prevent future assets from becoming defect. This is very important since SPL is about long term reuse and correcting assets that have already been used in products can be costly.

Final Grade: ++

Process Area: Organizational Innovation and Deployment (OID)

Purpose: To select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies. The improvements support the organization's quality and process-performance objectives as derived from the organization's business objectives.

SPL Comments: This is of course an important process area although we can not think of any SPL specific situation when this would more or less important than for single system development.

Final Grade: 0

14 Appendix H: Lessons learned in case studies

Lesson Learned	Description	Source
Re-engineering of legacy products is an effective approach when the products remain similar.	Reengineering when the products are similar have been proven time saving since there are not that much to change and refine.	[KIM07a]
Design guidelines is a handy tool for developers.	If the developers follow well structured generic guidelines, then the products have higher chance of being generic as well and thus have a longer lifetime.	[KIM07a]
Scoping is one of most essential but difficult activities in the creation of a SPL.	Getting the scope correct is crucial because otherwise you will develop the wrong products.	[KIM07a, KOS06]
Senior management support is critical.	Without support from senior management in form of funding and direction.	[JEN07, KOS06]
A product line architecture is essential.	Only basing the SPL on legacy products is risky and is likely to fail.	[JEN07]
Address product line requirements on support tools and processes first.	If the tools and processes are not adequate for SPL development then it becomes hard to use SPL which might lead to frustration and poor products.	[JEN07]
Determine the process for domain analysis that best fits your organization and put the processes in place to perform domain analysis activities as early as possible	This is to get the domain correct with minimal effort and to have the domain ready when planning the rest of the transition process. It also ensures that all project participants agrees on the concepts of the SPL.	[JEN07]
The transition in product line practice areas such as infrastructure, core assets, organizational structure and development processes should not occur in parallel.	Doing the transitions in sequential order made it clear after one step was complete, what had to be done next and why.	[HET06]
Have quick and continuous return of investment.	This makes it much easier to quell the disbelievers.	[HET06]
Having early and continuous communication of SPL concepts.	This will increase the commitment of the middle management and the developers.	[STE04]
Using traceability for knowledge communication.	This enables the transfer of knowledge to users or others interested in learning about the systems.	[LAG04]

Automating product derivation.	It needs automation to better discover inconsistencies.	[LAG04]
Integrating architectural information and the feature map in the SPL.	To get an improved overview and to spot inconsistencies between features and architecture.	[LAG04]
The software architect must be in place and have clearly defined instructions and priorities.	To be able to create an generic and usable architecture.	[KIR06]
Separating the problem space and solution space when creating the requirements as well as the design.	To simplify the development of requirements and design and make them more consistent and concise.	[KIR06]
The organizations need to attain a certain level of maturity before initiating SPL.	This is to have sufficient processes that can handle the SPL development.	[KIR06]
Having an well constructed mechanism for implementing and handling variability.	Variability is a important part of SPL and must be handled in an appropriate way.	[KIR06]
Close collaboration between product management and development	The monitoring of the developer by the management should lead to more generic and reusable core assets.	[KIR06]