CHALMERS

Software Product Line Engineering Maturity Model for Small and Medium Sized Organisations

Master of Science Thesis in the Programme Software Engineering and Technology

SIIM SAARLO

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, June, 2009

Software Product Line Engineering Maturity Model for Small and Medium Sized Organisations.

S.Saarlo

Examiner: T.Gorschek

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June, 2009

Abstract

This work is an attempt to create a software product line engineering maturity model (SPLEMM) suitable for small and medium sized enterprises (SME). Existing frameworks were analyzed, restructured, reduced and populated with examples from available case studies. Theory based model was validated and expanded with findings from empirical study.

Result of the work is the maturity model that provides a way to get an overview of SPLE paradigm and necessary adoption activities for potential SPLE implementers. The model can be also used to evaluate and compare existing product line activities, and to plan and support improvements in SPLE processes.

Based on the conducted study it was recognised that available theory lacks proven case studies involving practitioners of SPLE. That is why there is a lot of room for expansion of the model in form of example activities.

Acknowledgements

Foremost I want to thank my supervisors Tony Gorschek and Martin Ivarsson for motivating me and helping through situations when I was stuck. Tony Gorschek for providing very fast feedback and concrete, directive expertise. Martin Ivarsson for contributing into discussions that always lead me some important steps further.

Also I am very greatful to people in Syntronic [7], Carmenta [1], TIBCO Spotfire [6] and Lavasoft [3] for finding time to provide me with all so valuable information.

Contents

List of Figures

List of Tables

1 Introduction

SPLE (software product line engineering) is a software development paradigm with a growing popularity in industry and academy as a method which brings benefits to organisations that develop multiple products in one domain. Developed products share number of commonalities but also have differences. By developing those products from a common set of core assets in a prescribed way, and managing commonalities and differences between them, it is possible to gain remarkable improvements in development costs, time to market, quality and productivity. [5, 31]

SPLE is mainly practiced in big organizations for developing embedded systems [25]. Still there are no evidences suggesting that SPLE is not suitable for SMEs (small or medium sized enterprises). Instead descriptions of successful adaption cases in SMEs can be found [25, 31, 30, 37, 43]. These case studies allow to make an assumption that with appropriate adaption techniques it is possible to successfully implement SPLE theories in smaller scales and reach the benefits that have made the paradigm so popular in big organisations.

Although adopted in some SMEs, there are no guidelines focusing on small scale SPLE adaption. Some form of guiding "cookbook" into SPLE is necessary for the theory to gain wide spread [24]. Available frameworks and pattern collections tend to be complex and lack the approach suitable for small scale practitioners [11].

Most of the available research in the field of SPLE is directed towards the initiation of a product line and its evolution does not have so much attention [39]. Organisations need a model that offers guidance for sustainable evolution of development processes subsequent to initiation of SPLE practices.

Current work addresses these issues by creating software product line engineering maturity model (SPLEMM) that takes into consideration the needs and restrictions faced by SMEs. This is done by gathering and analysing available research in the area and organising it in the format suitable for target audience. The model is organised both into process areas and maturity levels. These principles allow the final model to be used by practitioners to get a thorough overview of SPLE, evaluate the state of SPLE practices, benchmark, start adoption process, and further develop already existing SPLE processes. The model is based on analyse of existing theory and state of the art case studies, and is validated in industrial context.

1.1 Disposition

Rest of the document describes five main topics: theoretical framework, project methodology as it was planned, outcome of the project, validation of the model, and final conclusions. SPLEMM is presented as an appendix of the document.

In theoretical overview main subject related concepts are presented. Issues that were found as lacking in the current state of theory are summarised in the end of chapter 2. Chapter **??** presents the research question that are motivated with missing parts in the theory. Also the methodology of the project, that was used to answer the research questions, is described. Developed model is thoroughly explained in chapter **??** and organisation, results and analyses of empirical part of the project are given in chapter 5. The conclusions of the project are presented in chapter 6.

Theoretical framework

This chapter gives an overview of main theoretical concepts related to the project. High level introduction into software reuse is provided and followed with main concepts of SPLE. Overview of existing SPLE related research in the fields of SMEs and existing maturity models are given as well. Chapter is concluded with a summary of current state of the research and aspects that were found missing in current theory.

Method

This chapter specifies the main principles and methodologies of the project. Research question is formulated based on the deficit in theory. The question is thoroughly explained in the beginning of the chapter. The planning of two main parts of the project - building the model based on theoretical research and validating the model in industry - are explained here as well.

Presentation of SPLEMM

This section presents the logic and principles of the model developed during the project. Structure of the model is described in chapter **??**. Mandatory and optional elements of the model are defined with the description of the hierarchies they are organised into. Chapter **??** explains the maturity levels and rest of the chapter **??** explains the usage and limitations of the model.

Validation of SPLEMM

Chapter 5 gives an overview of conducted validation activities in industry. Overview is provided about planning of validation interviews, interview outcomes and analyse of results. Main findings about each maturity area are given in chapters 5.3.2 - 5.3.5. Number of modifications were made in the model after analyse of interview results. These changes are listed in chapter 5.4.

Conclusions

Final chapter summarises the the project by listing the learnings and main experiences. Research question is reviewed again and mapped to the findings of the project. Finally some suggestions for future research directions are proposed and whole work is summarised in a brief manner.

2 Theoretical framework

This section gives an overview of available theoretical framework that the project is based on. Subsections introduce important concepts and references that had an influence on development of SPLEMM. Section 2.1 gives brief overview of general software reuse concept, section 2.2 introduces the concept of SPLE and some related aspects. Sections 2.3 and 2.4 describe software development and SPLE in the context of small and medium sized enterprises. SPLE connections with agile methodologies is discussed in section 2.5 and section 2.6 gives and overview of some more common models and frameworks studied during the project. Theoretical part is finished with a section 2.7 which summarises the issues that by author's opinion have not been studied enough in state of the art theory.

2.1 Software reuse

Following section gives a brief overview about general aspects of software reuse, involved methodologies and differences between SPLE and other reuse methods.

Krueger [26] has defined the software reuse as the process of creating software systems from existing software rather than building software systems from scratch. This gives an effect of reduced effort required to build software systems. As quality of software systems increases due to reuse of quality artifacts, the maintenance costs decrease as well. [26] The idea of software reuse is as old as programming [19, 30]. The earliest researched approaches in the field were reusable components, program families and concepts of domain and domain analyses. Other actively studied areas listed by Frakes et al. [19] include reuse libraries, domain engineering methods and tools, reuse design, design patterns, domain specific software architecture, componentry, generators, measurement and experimentation, and business and finance. [19]

Abstraction is stressed as the main requirement for success of software reuse by Krueger. Efforts in this direction has been rewarding both in industry and in academic research. Example of high level abstraction is the settings of reusable artifacts where developer can choose artifacts based on "what they do" instead of being forced to deal with "how they do it" questions. [26]

So called small-grained reuse [30] bases on reuse of libraries containing algorithms, modules, objects, or components. Major disadvantage with this kind of libraries is that it takes often more time for a developer to find and integrate a suitable artifacts from there than to build one from scratch. Industrial experience have shown that reuse under these settings is fortuitous and payoff is usually nonexistent. [30] Compared to this paradigm SPLE has more planned, comprehensive and profitable approach to reuse. The asset base includes also other artifacts besides code. Namely requirements, domain models, architecture, test cases and components - the ones that are most costly to develop from scratch. [30]

SPLE can be compared to component based development. Although the products on product line are based on reuse of components, these components are all defined by product line architecture and assembled in prescribed way. Prescription of derivation procedure includes the built-in variability mechanisms in the components. SPLE can also be distinguished from reconfigurable architecture and set of technical standards as its approach to reuse is much wider.

[30]

2.2 Software Product Line Engineering

Software product line engineering (SPLE) is a paradigm of software development with a growing popularity that aims to minimise rework and maximise reuse. The paradigm is suitable for organisations that are developing several software products in one domain. Using the commonalities and managing differences it is possible to develop these products from a set of predefined core assets in a prescribed way. Concentrating on full product line and reuse through customisation allows organisations to gain remarkable benefits in a form of reduced development costs and faster time to market, increased quality and reliability of products. [30, 31]

As examples of more formal definition can be given one by Pohl et al. [31] :

> Software product line engineering is a paradigm to develop software applications (software-intensive systems and software products) using platforms and mass customisation.

and alternative by Clements et al. [30]:

> A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

2.2.1 History

The idea of minimising rework through reuse is not new in software engineering. Most of the initiatives were based on small-scale ad hoc code level reuse. Development of reusable assets focused on specific domain has been practiced for some time in form of automatic code generation in a single domain. This has lead to domain specific languages. [42]

Something similar to product line concept was first proposed by Parnas in 1970s as product families [42]. Term software factory, which is used as a synonym to product line, goes bit more back in time. It was first used by R. W. Bremer of General Electrics who explained the term through focusing on standardised tools and controls. In 1968 M. D. Mcllroy at AT&T stressed systematic reuse of code on development of new programs. Practical usage of the term started in Japan where Hitachi company used term kojo (factory in English) to label its software development facility in 1969. But it still took some time for the concept to mature. [22, 16]

Precise concept was developed in the early 1990s. Initiatives like Feature-Oriented Domain Analysis (FODA) method and systematic approach from companies like Philips leaded to further development of the paradigm in both research and industry. In Europe there were many scientific groups: Architectural Reasoning for Embedded Systems (ARES 1995-1998), Product-line Realisation and Assessment in Industrial Settings (Praise 1996-2000), Engineering Software Architectures, Processes and Platforms (ESAPS 1999-2001), from Concepts to Application in system-Family Engineering (CAF 2001-2003), FAct-based Maturity through Institutionalisation, Lessons-learned and Involved Exploration of

System-family engineering (FAMILIES 2003-2005). At the same time in USA the subject was lead by Software Engineering Institute (SEI). [42]

2.2.2 Fundamentals

The main difference between single system development and SPLE is that the focus is shifted from the single product and project to the product line. Strategic vision covers the whole field of business in specific domains instead of concentrating only on next projects and contracts. [42]

Product line engineering consists of two life-cycles: domain engineering and application engineering (Figure 1). Due to their purpose they can be described as development for reuse and development with reuse respectively. Domain engineering develops the assets that can be later used in product development life-cycle. When other reuse approaches focused on reuse of coding assets then domain engineering in SPLE prepares reusable assets for product development throughout whole life-cycle from requirements until testing. Application engineering depends strongly on the assets provided by domain engineering. Applications are derived from reference architecture using available core assets and prescribed methods. Different projects may have up to 90% of a new product available in core assets. [42]

Successful product line engineering relies on fundamental principles [42] :

- Variability management: individual systems are considered as variations of a common theme. This variability is made explicit and must be systematically managed
- Business centric: Software product line engineering aims at thoroughly connecting the engineering of the product line with the long-term strategy of the business
- Architecture-centric: the technical side of the software must be developed in a way that allows to take advantage of similarities among the individual systems
- Two-life-cycle approach: the individual systems are developed based on a software platform. These products - as well as the platform - must be engineered and have their individual life-cycles (Figure 1)

2.2.3 Benefits

Success of SPLE paradigm relies on the presumption that accumulative cost of developing reference architecture and deriving new systems from that is lower than creating each product from scratch. It can be expected that creation of a set of core assets, that includes common parts of product line systems and takes into account the variations between them, is relatively complicated task compared to development of single system. Consequently development of reference architecture and deriving only one product from it costs more than building single product from scratch. Thus it is justified to ask when starts SPLE to pay off. By Linden et al. [42] investments into SPLE start making economical sense when core assets are used to derive at least three products (Figure 2). This can be seen as a point when sum of costs for developing each

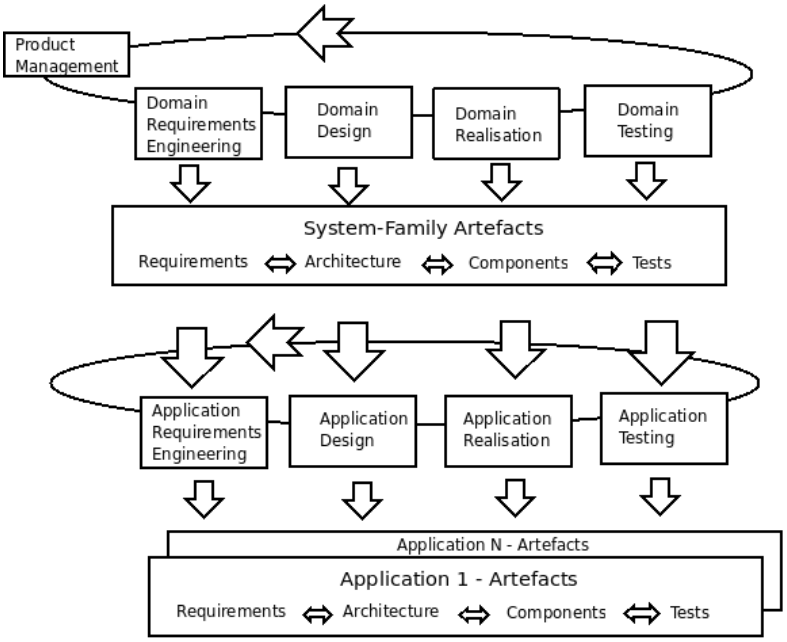

Figure 1: The two-life-cylce model of SPLE [42]

system from scratch outweighs investments needed for first developing reference architecture and then deriving same products from there. [42]

Besides lower development cost it is possible to achieve better time to market with new products derived using SPLE practices. [42]. This affects organisations ability to deliver new products or versions of products to customer fast and regularly. This feature can play important role in success and failure of product development organisation in highly competitive markets.

Minimising the work to be done with each single product allows to lower the risks of development errors. As products are largely based on reuse of thoroughly tested core assets, reliability of these assets is transferred to produced systems. [42]

Other improvements recognised after SPLE adoption are reduced code size through removing duplicated code, satisfaction of people involved in development, more efficient use of human resources, increased market agility, ability to effect mass customization, reduced maintenance costs and common user interfaces through products. [5, 42]

2.3 Software development in small and medium sized organisations

European Commission categorises enterprises by their headcount and turnover. Micro enterprises have headcount below 10 and turnover up to 2 million euro. Small enterprises are the ones with less than 50 people and up to 10 million euro turnover. Medium sized enterprises have headcount below 250 and turnover up to 50 million euro.[2] So when designing a SPLE maturity model for SMEs we are considering software development companies with less than 250 employees

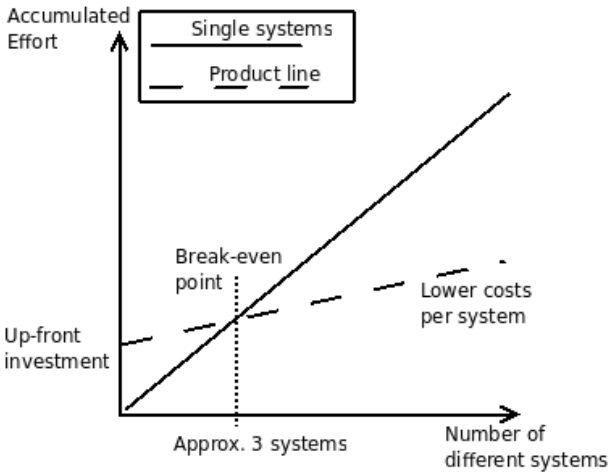

Figure 2: Economics of SPLE [42]

and up to 50 million euro turnover.

Tore Dyb have made a study among Scandinavian small and big sized companies to analyse the differences in practices and results of their software process improvement efforts. It was recognised that small size does not limit organisation's potential for SPI success. When properly utilizing its strengths, small organisations were able to implement SPI as effectively as their bigger counterparts. High employee participation and exploration of new knowledge and possibilities were recognized as critical success factors for effective SPI in small scales. This requires effective mix of formal practices and informal, interpersonal coordination from small sized practitioners of SPI. Dyb concludes that software processes for smaller companies should support different formality levels: "a general description of the process, supplemented by specific examples from practice." [18]

There are several problems affecting process improvement activities in small sized companies. SMEs often bare under limited resources and strict time constraints [34, 44]

In most cases they can not afford to maintain SPI expertise in house. Instead they have to buy it in from external resources. Also it is difficult to find out how to start the improvement and what experts to use [29]. As SMEs often lack formal processes [44, 25] Wangheim proposes descriptive modelling as a suitable approach for process improvement in SMEs. This is a practice of describing as-is processes and then completing or changing process elements with suitable best practices where necessary. [44] Smaller organisations have a structure that encourages innovation. It is flat and lead by organic management [34].

2.4 SPLE in small and medium sized organisations

SPLE is more common in bigger organisations that can afford and manage the long term planning, process changes and other investments needed for the practice of the paradigm. Smaller companies often have the flexibility and agility as competitive advantages [25, 34]. This makes it seem like SPLE and

small organisations do not make a good combination [25].

Number of case studies have shown that the benefits SPLE offers - lower development costs, decreased time to market, increased quality - are also achievable when applying the theory in smaller organisations. Small size can even be a benefit in SPLE adoption. This often means that less rigid organisational structures and processes are sufficient and troubles of distributed development can be omitted [12, 43]. Also when talking about the spread of SPLE among smaller companies, then it has to be noted that lot of companies already have SPLE practices without being aware of it [10]. Following gives a brief overview of example cases that were studied during the project.

Alves et al. have described the exploratory study of SPLE practice in SME active in the mobile games domain. The work analysed the main challenges in the domain and how SPLE practices could help solving those. Studied company faced some complications because of the relatively new and complex domain but overall adoption project was considered successful. It was concluded that although somewhat complicated to adapt, SPLE is still suitable approach for SME-s in mobile games domain. [10]

Bosch has studied SPLE experience in two Swedish companies Axis Communications AB and Securitas Larm AB from the perspective of product line architecture. It has to be noted that these companies are not exactly fitting into SME definition. The case study is still used as theoretical base because of its suitable focus and research direction. The article identified several problems like limited background knowledge, information distribution, asset versioning, use and dependencies between assets, documentation, tool support, management involvement and effort estimation. [13]

Reactive software product line approach in start-up company is evaluated by Buhrdorf et al. Studied company achieved remarkable benefits in a very short time of adoption. The study showed how the risks of uncommon domain and upfront investments can be mitigated with usage of reactive adoption processes. Several other good case practices for SPLE adaption were pointed out as well in the article. [36]

Birk et al. have studied the experience of Market Maker as one of the 6 cases, when doing the research about SPLE practices [12]. The same company's case was more thoroughly evaluated by Verlage et al. who have followed the successful practice throughout 5 years and pointed out the lessons learnt from the experience. The company was able to gain benefits from the stable domain of stock markets and hence scoping activities are stressed in report. [43]

Another example of SPLE in SME was mentioned by Deelstra et al. who used Dacolian BV for testing their variability assessment technology. The company was described as a successful practitioner of SPLE. [17] An alternative case study with Spanish SME follows company's SPLE adaption process through IDEAL steps presenting the original context and motivation behind transformation, migration process, gained benefits and learning (REF: Clements et al). Knauber gives an overview of a research project that aimed to transfer product line engineering concepts to SME-s using and adapting Pulse methodology. The project was initiated by IESE in 1997, involved 6 companies, and lasted for 2.5 years.

Similar issues can be recognised throughout the different experiences with SPLE in small and medium sized companies. These issues can be taken into consideration as requirements or principles when creating maturity model for

SPLE. Described SPLE adoptions were successful and achieved the aimed benefits but it can be noticed that due to a lacking maturity in processes the gaining was sometimes not exactly measurable. This explains the importance of management support which was pointed out as a crucial success factor in SPLE case studies with SMEs [37, 25]. More abstract or invisible are the benefits in early stage the more it has to be guaranteed that key decision makers understand the long term winnings and background behind the SPLE adaption efforts.

SMEs seldom have well defined processes set in company. This complicates SPLE adoption as it is hard to change non-defined processes and non-existing baselines [25]. On the other hand small size of organisation allows to mitigate the importance of organisational structure and rigid processes. Implementation of rigid processes can be too costly for smaller enterprises [13]. As there is a lack of available tools supporting SPLE needs, it is suggested to invest into development of own tools. These tools can replace alternative systems of papers and processes. [13, 37] Although light weight methodologies are more suitable for SME-s proper adaption of rigid processes can bring remarkable benefits to smaller companies [25, 43]

Important issue to consider in SME cases is usage of flexibility as a competitive advantage and SPLE influence to it. When SMEs are often very responsive to customer demands then SPLE practicing needs confidence about future product developments. Prioritisation and compromises are used to overcome this conflicts in available cases. [43, 12, 25]

Importance of using legacy systems on core asset development was stressed in studied SME cases. Proper component extraction and encapsulation can crucially speed up SPLE adoption in SME. [36, 43, 25] Also it was common that organisational division into application and core asset development teams was avoided [43, 12]. Lack of domain knowledge was recognised as a often occurring problems in new domains and small sized companies [10, 25]. Information gathering from more experienced stakeholders was used to mitigate this issue [37, 25]. Also extractive and reactive adoption strategy where SPLE is adopted gradually helps to deal with the problem of lacking data [10].

2.5 SPLE and agile methodologies

Usage of agile methodologies is widely popular in software development specially in small scales [33]. Knowing that it is much easier to modify existing processes instead of changing them totally [44], it is justified to ask how SPLE and agile methodologies fit together.

Tian et al. have studied the question more thoroughly through the perspectives of engineering, software quality assurance and project management [40]. When SPLE and agile methodologies share the same high level goal to deliver quality software quickly, they have different strategies for achieving it [40]. Similar observations are made by other authors too [32]. But as the differences are not fundamental, several studies propose to tailor SPLE with agile practices [40, 32, 28, 23]. Carbon et al. successfully added agile practices like planning game, continuous integration and automated regression testing into PuLSE-I product instantiation processes [32]. Adaption of agile practices into product line planning activities are also studied in couple of works [28, 27]. In work by Tian et al. it is proposed that agile methodologies may be good addition to SPLE when there is not enough knowledge for thorough SPL scop-

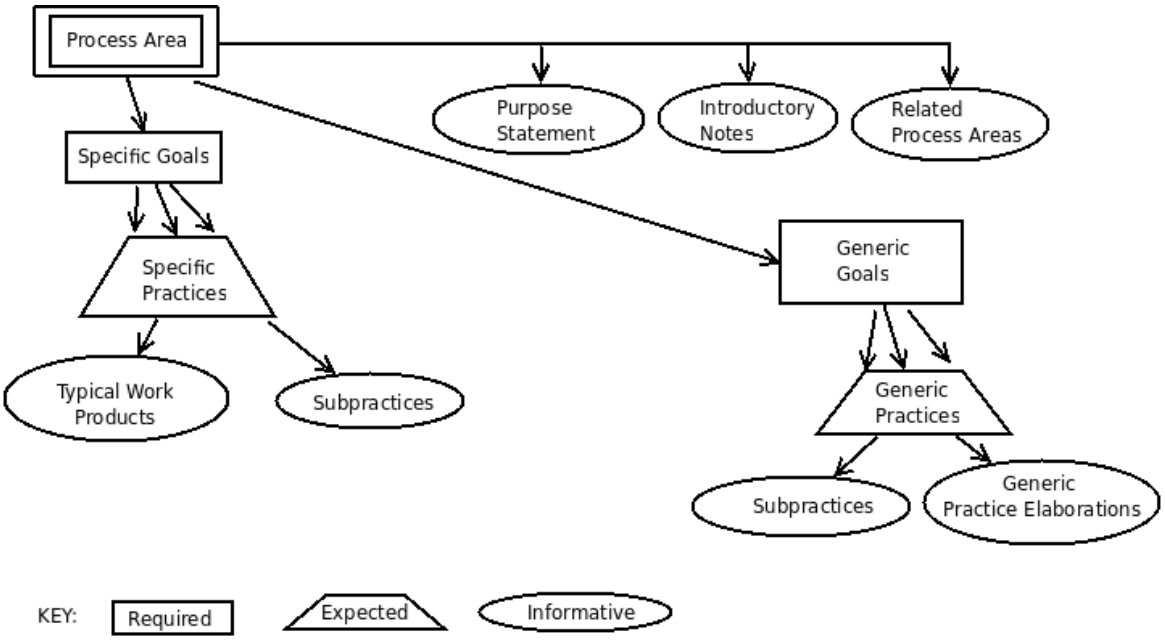

Figure 3: Components of the CMMI model [38]

ing. Quickly built and easily modifiable prototypes make it possible to manage changing scope [40]. Still making SPLE agile should not be the goal itself but in some cases agile methodologies can add some flexibility and reduce complexity in SPLE. [40]

2.6 SPLE maturity models and frameworks

This section gives a brief introduction into existing models and frameworks that were used as example when developing SPLEMM. First CMMI is introduced, which is not SPLE specific but covers whole software engineering area. Later overview about SPLE related works is given.

2.6.1 Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) is probably the best known process improvement model for software engineering. It was developed from Capability Maturity Model (CMM) that was introduced by Carnegie Mellon University's Software Engineering Institute already in 1980s. According to them CMMI is a process improvement approach that provides organisations with essential elements of effective processes. Its application areas include process improvement and assessment across a project, a division, or entire organisation. [4]

CMMI consists of 25 process areas divided into four groups: Process management, Project management, Engineering and Support. Each process area includes set of specific goals and generic goals, that are considered as required components of the model. Specific and generic goals organize specific practices

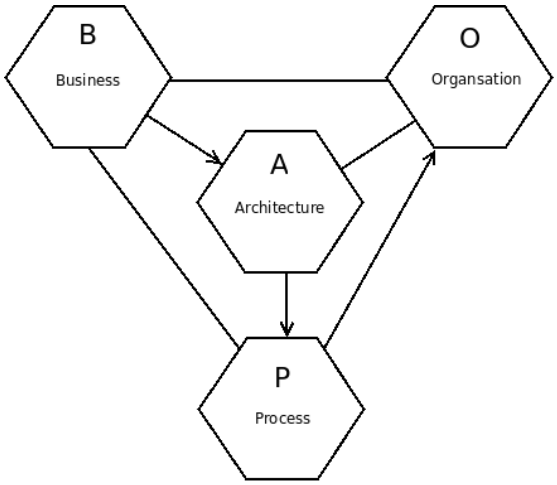

Figure 4: BAPO concerns of SPLE [42]

and generic practices respectively. These are expected components of CMMI model and should be performed in an organization to achieve a concrete goal. Practices contain subpractices and other helpful hints or guidelines which represent informative part of the model. Specific goals are unique for each process area but generic practices apply to multiple process areas and describe the level and quality of performance in that area. Only required elements are necessary to achieve a maturity level in CMMI. (Figure 3)[38]

Continuous representation of CMMI, that is more flexible than its counterpart staged representation, has 5 maturity levels: Incomplete, Performed, Managed, Defined, Quantitatively managed, and Optimizing. Organization can choose to work and improve single process area or simultaneously develop performance in multiple areas, when they use continuous representation of CMMI. It is also possible to improve different process areas in different rates. [38]

2.6.2 Family Evaluation Framework

Family Evaluation Framework (FEF) is a framework for assessing organisations effectiveness in SPLE and can show the possible improvement opportunities in the area of SPLE. It only includes areas specific to product line engineering and leaves out general software engineering issues. FEF is divided into four BAPO concerns: Business, Architecture, Processes and Organisation (Figure 4). Each area has number of aspects and defined levels of maturity. This makes it possible to measure level of organisation in each concern separately. So the final result of FEF evaluation gives a result in form of four values - one for each BAPO area. FEF has taken over a lot from CMMI. The framework has five maturity levels with first one being achievable to everyone. FEF also follows the CMMI practice of giving guidance of what should be done for each level instead of explaining how should it be done. FEF still is less comprehensive as it does not specify the components like goals, practices and work products (Figure 3). [41, 38]

Business dimension concentrates on the issues that are relevant from the

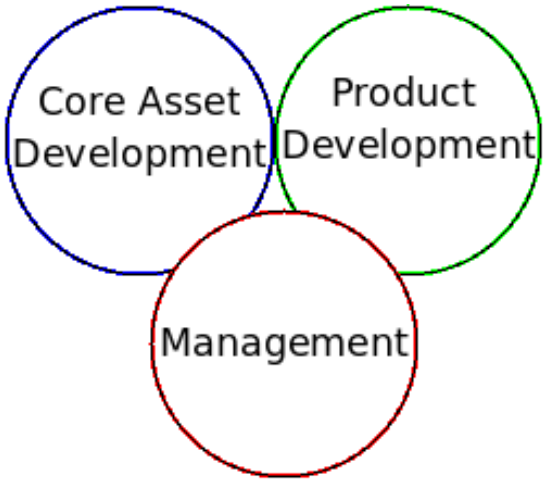

Figure 5: Essential activities of SPLE in SEI Framework for Software Product Line Practice [30]

perspective of how to make profit from developed products and considers the business relationships between domain and application engineering. The dimension is divided into four aspects: Sales Marketing and Product Management Involvement, Budgeting and Investment, Vision and Business Investment, and Strategic Planning. Maturity levels for business dimension are Project Based, Aware, Managed, Measured, and Optimised. [41]

Architecture dimension concerns with the technical means to build software. The dimension consists of Asset Reuse Level, Software Product Family Architecture, and Variability Management aspects. Maturity levels are named as Independent Product Development, Standardised Domain Independent Infrastructure, Software Platform, Derivable Variant Products, and Automated Product Derivation. [41]

Process concern area deals with roles, responsibilities and relationships withing software development. It is divided into Domain Engineering, Application Engineering and Collaboration aspects. Maturity levels under this concern area are the same as in CMMI: Initial, Managed, Defined, Quantitatively Managed, and Optimising. [41]

Finally Organisation concern area deals with the actual mapping of roles and responsibilities to organisational structures. It consists of Roles and Responsibilities, Structure, and Collaboration Schemes aspects. Maturity levels are Project, Reuse, Weakly Connected, Synchronised, and Domain Engineering. [41]

2.6.3 SEI Framework for Software Product Line Practice

The SEI's Framework for Software Product Line Practice sees the essence of software product lines in interconnection of three essential and iterative product line activities: Core Asset Development, Product Development, and Management (Figure 5). [5]

The framework itself describes 29 practice areas from different aspects of

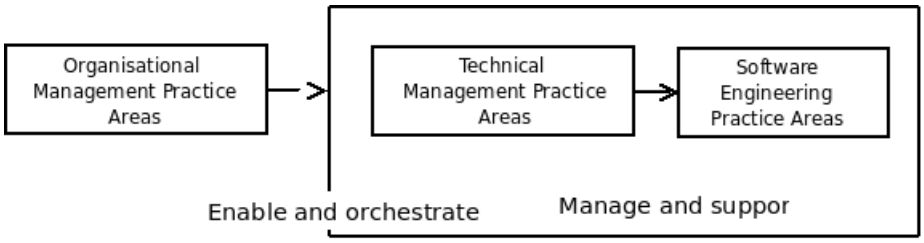

Figure 6: Categories of process areas in SEI Framework for Software Product Line Practice [30]

SPLE. Practice areas are organised under 3 categories: Organisation Management, Technical Management and Software Engineering (Figure 6). Practice areas in the same category share a relation that might be described as "requires the same kind of knowledge as" or "requires the same kind of skill set as". Software Engineering practice areas deal with the question of how to apply the technology to create and evolve both core assets and products. Technical management is needed to engineer the creation and evolution of core assets and products. Organisational management practice areas synchronise all SPLE activities. [5, 30]

Practice areas in the framework are quite thoroughly described. Besides general introduction each area is equipped with a list of aspects that are specific to SPLE. Practice area's relation to both Core Asset Development and Product Development essential activities are also given. The framework additionaly introduces the example practices that are given as possible ways to deal with the problems in concrete process area. [5]

SEI also offers a different organisation of practice areas. Software Product Line Practice Patterns act as a guidance that allow implementation of knowledge that is gathered into separate practice areas. These patterns are collections of practice areas that are organised according to the most common needs of SPLE practitioners. Software Product Line Practice Patterns are modified to be applicable for concrete situations and parts of the product line effort. Each pattern is described with example scenarios, contextual overview, problem statement, offered solution and few other attributes. Patterns described by SEI include The Curriculum Pattern, What to Build Pattern, Cold Start Pattern, Factory Pattern and many others. [5, 30]

2.6.4 PuLSE

PuLSE is a methodology for developing and deploying software product lines. It was created as an effort to target the shortcomings present in other domain engineering approaches - deployment complexity, lack of customizability, and misplaced focus on domains. Instead PuLSE is stressing on a product centric focus, component customizability and incremental introduction capability. The methodology gathers the learning from industrial cases facilitated by Fraunhofer Institute of Experimental Software Engineering. [11]

PuLSE methodology is organised into three main elements: deployment phases, technical components and support components (Figure 7). Deployment phases are logical stages that product line goes through. Initialisation stage in-

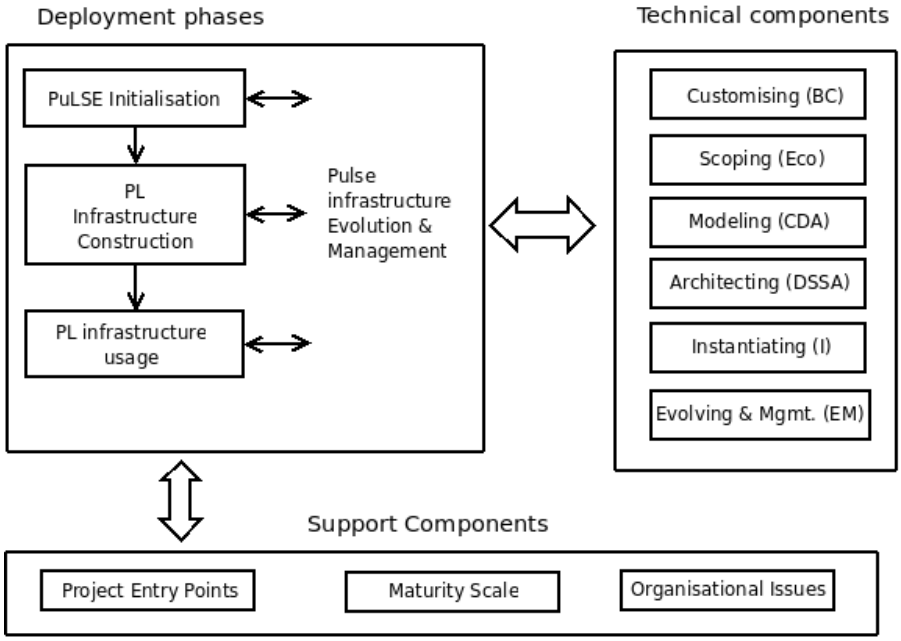

Figure 7: PuLSE Overview [11]

volves base-lining the organisation and customising PuLSE methodology to its needs. In Infrastructure Construction stage scoping and modeling is done and product line infrastructure is architected. Product line members are created from Infrastructure Usage stage and finally Evolution and Management stage deals with evolution and management of product line over time. [11]

Technical components, that were developed before other components, provide necessary technical expertise to certain deployment phases. Customizing (BC) provides the solution for Initialization phase. Also there are Scoping (Eco), Modeling (CDA), Architecting (DSSA), Instantiating (I) and Evolving and Management (EM) technical components provided by the model. [11]

The three support components in PuLSE methodology are guidelines used by other elements to support better adaptation, evolution and deployment of a product line. Project Entry Points customise PuLSE to major project types. Maturity Scale provide evolution path for adapting organisations. And Organisational Issues support component provides guidelines for organisational structure of product line organisation. [11]

2.6.5 Maturity and evolution in software product lines

Bosch have defined a set of maturity levels to describe the common stages of SPLE evolution. Initial maturity level is standardised infrastructure where common operating system with components on top of it. Third party domain specific components may also be required. Further development goes through platform stage where common infrastructure is implemented by the organisation. Next, on software product line maturity level, shared artifacts include the functionality that is common to several, but not all, products. Configurable product base

concentrates on automating the product derivation process from core assets. More advanced levels are program of product lines and product populations which can be useful to target in large scale product line engineering.[14] The article gives rather general overview about different stages product SPLE can go through and is not usable as independent model or comparable to previously described frameworks.

2.7 What is missing in available theory

Current state of research is missing a source of reference specialised for software product developers in small and medium sizes that have some principal differences compared to their bigger counterparts. Author's opinion is that existing guidelines and frameworks for SPLE adaption and evaluation seem to be complex and need a lot of overhead to adapt for a concrete case. From the other hand available case studies that give an overview of a concrete instance of SPLE usage are hardly extensible for wider audience. Compromising tool could be welcomed by practitioners. This tool should combine the high-level overview of SPLE paradigm provided by existing models with concrete steps necessary for successful implementation of the SPLE.

Many organisations are already using practices from the SPLE paradigm without calling it product line engineering or connecting it with the theory. It can be assumed that this is more of a issue with SMEs as they probably have less overview about different state of the research theories and invest less into SPI activities compared to big corporations working with software product development. Unification of separate procedures into one thorough theory brings in obvious benefits. Recognising some parts of the SPLE already used in an organisation may have negative effect on further interest of SPI. First seeing that parts of the paradigm are already done in an organisation could rise doubts about possible added value from the paradigm. Also seeing that the used tool does not provide easy and understandable proposals but need lots of analyse and investment could dampen the excitement drastically. There is a need for a tool that gives an overview of companies current level and possible advancement opportunities fast and clearly. This would motivate SMEs to continue with SPI initiatives.

3 Method

This section presents the research question of the project. Also general overview is given about the approach and structure on how the answer for the question was searched.

3.1 Research question

The main research questions of the thesis project is:

> **What is the suitable software product line engineering maturity model for small and medium sized organisations?**

The model should provide following functionality for its target group:

- **Overview of SPLE** - Organisations using the model should get a good overview of SPLE paradigm. Besides having a definition of the paradigm it is important for companies to know what does the paradigm involve in form of actions and processes. Lot of research and work on the subject should not be needed but initial understanding of the paradigm and related activities must be easily conceived.
- **SPLE adoption** - The model should support first time adoption of SPLE. Companies who want to start using SPLE can use the model to build up their own SPLE processes. It must provide overview and definitions of process areas in the paradigm plus some concrete references on how to work in the necessary fields. SMEs might not have the resources or will to start with something that consists of only abstract overview. Thus some form of concrete activities has to be included in the final model.
- **SPLE process improvement** - Organisations that are already using SPLE should be able to use the model for assessing and improving their processes. The model should provide opportunities for process assessment that allows organisations to clarify their current state in SPLE practices. Further it has to provide them with guidelines on how to improve their situation.

To further dissassemble the research question, the terms in it has to be clarified. Terms *product line engineering*, *maturity model* and *small and medium sized organisations* of the main research question are already discussed under the theoretical framework (chapter

In current work suitable model has following characteristics:

- **Usable** - Suitable SPLEMM has to be usable by its target group. This means that organisations can use the final model as it is or adapt it in a simple and understandable manner. These requirements mainly affect structure of the model which must not be complicated but easy to follow and understand.
- **Relevant** - The content of SPLEMM has to be relevant for its users. It has to be analysed if all the SPLE related issues have the same importance for smaller organisations. Most of the available knowledge is currently

based on the experiences from larger organisations and the model is created based on this knowledge. Thus the content of the model has to be validated to include only relevant aspects of the SPLE to smaller software developers.

- **Evolvable** - It should be possible to extend the model by adding new information to it. The area of SPLE is fairly new and expanding. New case studies from SPLE practitioners could contribute a lot to the practice and based on the image gained through theoretical overview there is a fair amount of active research happening in the area. Thus future updates in the model are most likely required to keep the model up to date with evolvement of the state of the art theory and practice. So the model has to be stable and evolvable in a sense that its principles remain valid when the model is expanded.

- **Tailorable** - Model should allow changes and modifications so it is possible to adapt it according to different process environments. Chosen target group of software product development SMEs is fairly large. Thus the model has to cope with a large audience. Users with different conditions and starting states should be able to use the model in beneficial manner. This requires that the model clearly addresses organisations on different levels and is flexible enough so that organisations may tailor it the way suitable to their individual needs.

- **Useful** - The implementation of the model should return more benefits than it costs. Organisations are willing to invest into projects that they can profit from. Thus in order to gain potential popularity the model has to attract users with positive outcomes that oversize all the initial and operational investments.

Theoretical research coverage of SPLE in smaller organisations context will be thoroughly evaluated during the project. SPLE related theory will be gathered and analysed from the SMEs perspective. This gives an overview about current state of the research and practice in SPLE in SMEs. Process of assembling the model based on theory and following validation may give some hints for the further research needs.

3.2 General structure of the project

Current work is not the first time attempt to adapt SWD methodologies or models for SMEs. It has been done before for SPI practices [29, 34] and often the work has followed similar basic structure: scaling down current state of the art practices and packing them into suitable form for smaller scales. [29] This approach were chosen for current initiative as well.

The initial idea of the project was to create a software product line engineering maturity model (SPLEMM) that would meet the requirements presented in chapter **??**. The first part of the project was theoretical study where initial version of SPLEMM was created based on available theoretical sources.

The model was later validated during the empirical part of the work where it was presented to selected industry representatives for feedback. Qualitative

(flexible) design [35] was used for empirical part and semi structured interviews [35] planned for validation.

Validation interviews were also used as a source for further input to the model. The final version of SPLEMM was composed by analysing both theory and interviews' results.

3.2.1 Creation of SPLEMM based on theory

Different sources of information were worked through during the theoretical study. These involved more general groundworks on the SPLE, specific studies concentrating on some aspects of the paradigm, and case studies about SPLE adoption and usage. Also the picture of SMEs and main limitations and benefits they are facing was created through related theory and used on assembling process. The studied material included the works related to topics presented under theoretical overview, plus more specific articles were used to populate the model with relevant examples.

Several other maturity models and process frameworks in the field of software engineering were analysed to find the most suitable composition for project aims.

3.2.2 Validation of the model

The aim of the validation was to first understand how industry representatives perceive the model and also to modify it to be more suitable for the needs of target audience. Validation was planned in a form of semi-structured interviews with representatives from industry. To achieve the first goal it was necessary to introduce SPLE concept, applicability and benefits to the interviewees. This was done with a brief oral presentation in the beginning of the interview. Interviewees understanding of the SPLE also grew throughout the interview when different parts of the model were explained. In the end of the interviews interviewees opinion about the model was measured with a quantitative questions. These questions were asked to find out how well involved people had understood both the structure and the content of the model.

Main part of the interview went through each process area in the model. The purpose of that was to introduce the model to participants and ask their feedback about the content. Interviewees were asked to answer if the process is performed in their organisation. In case of negative answer short reasoning was expected. In case of positive answer interviewee was asked to name couple of most important critical success factors that contribute to the success of the SPA in their organisation.

Process areas that were not performed in interviewed companies were checked against the given reason and their purposefulness in the model was analysed again. Success factors that were pointed out by industry representatives were later considered as possible additions to the model.

3.3 Choice of organisations for validation of the model

Organisations were chosen as potential users of the model. Interviewed organisations were chosen by two main criteria. They had to develop multiple software products and have a development in a small or medium sized settings.

Although sometimes it is suggested to distinguish between small and medium sized SWD companies [34], this work does not separate them. Instead the definition is left indefinite targeting smaller companies and using the European Commission's definitions [2] just as guiding suggestions rather than rigid restrictions. Thus no concrete numbers were followed during company selection. Still all interviewed organisations classified as SMEs [2].

More important criteria on interview subject selection was their experience with software intensive product development and management. Although potential for SPLE was considered important factor, no formal tests were made to find out organisation's potential. Instead it was assumed that if an organisation is producing several software intensive products in one domain or using same base technology there is some potential for SPLE adaption.

4 Presentation of SPLEMM

This chapter gives an overview of the software product line engineering maturity model (SPLEMM) that was developed during the thesis project. Following subparagraphs describe the elements and general structure of the model, maturity levels, and notation that is used for the model documentation. Evolution of the model through the project is described in the chapter **??**. Section **??** gives the overview and examples of different ways to use the model and final part of this paragraph presents the limitation of SPLEMM that potential users should take into consideration.

4.1 Structure of the model

SPLEMM consists of two types of elements - process areas and example actions - that are organised in hierarchical tree structure. Current section describes these elements and how they are organised in SPLEMM.

4.1.1 Process areas

SPLEMM is a collection of processes of software product development that have effect on and are affected by SPLE paradigm in organisation that practices SPLE. The model can be seen as a collection of factors that are necessary for the success of SPLE practice. These success factors (called process areas in the model) were found through the analyse of available theory and case studies in the field of SPLE. The model is organised into a hierarchical tree structure where lower level process areas contribute to the success of their parent element. Depending on the positioning on the hierarchy levels the process areas are categorised as maturity areas (MA), process areas (PA) and sub-process areas (SPA). Note that the term *process areas* are used both as a general term referring to all three categories (MA, PA and SPA) and as a name of second category (PA). In rest of the document general term is written out and when referred to specific category then abbreviation PA is used.

On the highest level of abstraction the model is divided into four MAs: Business, Domain Engineering, Application Engineering, and Collaboration. For an organisation to be successful in its general SPLE practice it is necessary to be successful in each MA. Each MA is divided into PAs which in turn contain

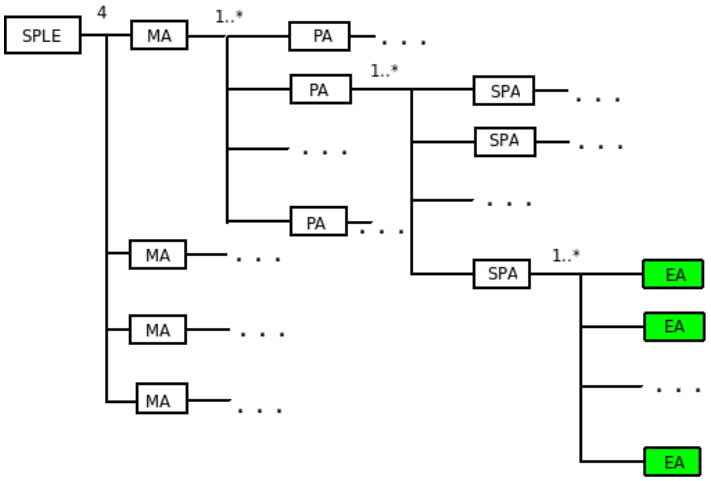

Figure 8: Structure of SPLEMM

SPAs. One SPA can be divided into other level of SPAs. Each SPA under one parent element contributes into success of its parent. It must be noted that process areas are divided into above mentioned categories only based on their positioning in the model. There is no other difference between the elements and all rules that apply to SPAs also apply to PAs and MAs.

4.1.2 Example actions

The model also includes example actions (EA). EA is an example of activity or method that have proven to contribute to the success of parenting process area. EAs are optional elements in SPLEMM. They are examples of possible ways on how to implement parenting SPA. EAs can be compared to informative components in CMMI [38]. EAs were added to the model from available industrial case studies, validated theories or interviews with practitioners. It has to be noted that collection of EAs under one SPA is not the only possible nor final list of actions to be performed under successful SPA.

Although SPAs and EAs are quite similar there still are some conceptual differences. EAs are more concrete and precise. When SPA-s answer question "what should be done" then EAs answer "how should it be done". For example SPA C.1.2.2 (*Mechanisms for Achieving Variability*) states that *Variability Management* PA should support *Domain Engineering* MA with knowledge about different variability mechanisms so that most suitable method could be chosen in different situations. EAs under this SPA are examples of concrete variability mechanisms that have been proven as useful (eg. *Using Application Specific Plug-ins* (C.1.2.2.ea1) and *Using Language and Generative Support* (C.1.2.2.ea3)).

4.1.3 Organisation of elements

As mentioned above the model consists of process areas (MAs, PAs and SPAs) and example actions (EAs). EAs are optional elements of the model or informative to use the terms from CMMI [38]. Process areas are mandatory elements

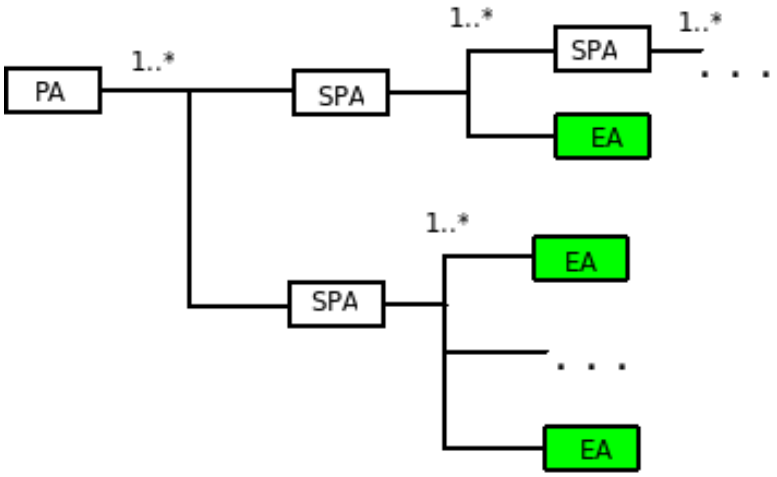

Figure 9: Example of SPA levels

of the model. Organisation has to satisfy all the process areas in order to be considered as SPLE practitioner (see section **??** for explanation of maturity levels).

Still satisfying a process area does not mean that an organisation has to perform it exactly as described in the model. In order to meet the requirement of tailorability (see section **??**) with SPLEMM the term *satisfied-explained* is taken over from the work of Gorschek et al. [21]. This means that organisation can satisfy a process area by not completing it but explaining that the process area in question is not applicable or not relevant in their context. Example of how satisfied-explained concept should be used is given in chapter **??**.

As explained before on the highest level of abstraction in the model are four MAs that all have to contribute to the successful SPLE performance in organisation (Figure **??**). Next level consists of PAs and each PA is divided into SPAs and/or EAs. Every SPA must have other SPAs and/or EAs as their sub-elements. Process area (MA, PA, SPA) can not be the last element in a branch. Instead the process area who have no lower level process areas connected to them have to have at least one EA below. EAs can not have any sub-elements (Figure **??**).

Business maturity area involves issues in the model that are directly related to business decisions. It can be said that this area answers to the question how to make profit with SPLE [41]. The MA is divided into three PAs: *Sales and Marketing* (B.1), *Scoping* (B.2) and *Business Planning* (B.3). *Sales and Marketing* PA has three SPAs that contribute to success of the PA. For example *Brand Name Strategy Aligned with SPLE* SPA deals with questions concerning brand strategy. EAs under it are *Defined Branding Strategy* (B.1.1.ea2) and *Brand Alignment With SPLE* (B.1.1.ea1).

Scoping is the process area that determines the products and product features which should be included in the SPL. It is divided into *Scope Planning* (B.2.1) and *Scope Maintenance* (B.2.2) SPAs. Some EA-s on lowest level of Scoping branch are *Studying Available Products in Existing Product Lines* (B.2.1.1.ea1), *Define Domain Candidates* (B.2.1.2.ea2), and *Developing Product*

Line Scenarios (B.2.2.1.ea1). *Business Planning* PA consists of *Strategic Planning* (B.3.1) and *Budgeting and Investment* (B.3.2) SPAs. Some descriptive EAs are *Managing Domain Engineering Budget* (B.3.2.1.ea2) and *Prediction of Future Cost and Benefits of Product Line Approach* (B.3.2.2.ea2).

Domain Engineering maturity area incorporates the process areas that control creation and maintenance of the reference architecture of SPL. These are mainly similar to processes in usual software development organisation. The MA includes process areas like *Domain Requirements Engineering* (D.1) and *Domain Testing* (D.3) among others. SPLEMM aims to present the differences between single-system software development and processes that run the successful SPLE practice. In the model representation these differences are often explained in the descriptions of SPAs or EAs. For example *Defining Production Strategy* SPA (D.2.1.ea1) is about choosing and using the variability mechanisms that is a SPLE specific concern.

Application Engineering maturity area involves the processes that control how products are derived from reference architecture and how modifications are made before the release of the final application. This MA as well follows the traditional software development life-cycle beginning with *Application Requirements Engineering* (A.1) and ending with *Application Testing* (A.4). But the essence of the maturity area is to reuse as much as possible from work done in Domain Engineering (eg. *Mining / Accessing Appropriate Assets* (A.2.2) and *Domain Test Artefact Reuse* (A.4.ea1)). The MA also deals a lot with using the options provided by variability (eg. *Binding of Variants* (A.2.3) and *Selecting Suitable Variants in Variation Points* (A.2.3.ea1).

Collaboration maturity area consists of PAs that run through both life-cycles of SPLE and thus should be presented under separate MA. *Variability Management* (C.1) manages the variability in domain architecture and how it is used by application engineering. *Configuration Management* (C.2) deals with the issues that make configuration management more complex in SPLE context. And *Organisation* MA (C.3) slightly touches the questions of skills and organisation of people in SPLE environment.

4.2 Maturity levels

The number of maturity levels is slightly lowered in the presented model. Compared to other approaches [38, 41, 9, 21] that distinguish between 5 to 6 different levels, SPLEMM defines 4. These are Non-performed level, Adaption level, Sustainable level, and Improving level.

Example actions in the model are mapped to certain maturity levels. The proposed mapping is based on the assumed cost and the complexity of actions but also on the relation of their estimated effects to the nature of each maturity level. Cost is seen as the measure of resources (money, time) that are needed to perform the action. Complexity is denoted as the extrapolated measure of how complex the action is. The higher the cost and complexity of activity, the higher is the level of its maturity. [21] The relation between EA and maturity area is more abstract and it needs some analyse to decide which is the level of maturity for new EA added to the model.

For example *Identify a Set of Common and Variable Requirements* SPA (D.1.2.1) has *Application-Requirements Matrix* EA (D.1.2.1.ea1) on maturity level 1 but *Priority-Based Analysis Scheme* EA (D.1.2.1.ea2) on maturity level

2. This distinction in the maturity levels is made because by author's opinion application-requirements matrix [31] is the simplest available tool for variability analyses and essential for light-weight SPLE practice. Priority based analyses [31] is more complex methodology but also gives more thorough overview about the issue.

Non-performed level (level 0) describes an organisations that do not practice SPLE or do it incompletely. 0 level organisation have not implemented all the mandatory parts of the model and do not have motivated alternatives for non-implemented areas. Organisations that have partial SPLE solution are often lacking SPLE essential high-level process areas like Variability management or sub-process areas on lower levels. There are no EAs mapped to level 0 in the model.

Adoption level (level 1) describes an organisation that is still in the middle of SPLE adoption process or has just finished it. The level mainly consists of product line essential actions that define the SPLE approach. Adoption level is necessary mid step for smaller companies who do not have resources for a full scale adoption in the beginning. It can be also considered by isolated parts of bigger organisations to test the suitability of SPLE practices. After achievement of this level efforts have to be made to reach sustainability in SPLE practices.

Level 1 organisation has to perform actions in all SPAs. Example actions under adaption level are simple and with low implementation cost. Often related to technical issues of SPLE building and initial organisation of support practices.

Sustainable level (level 2) provides organisation with mature enough processes to achieve sustainability in technological and economical development. Organisations on this level have looked forward from initial engineering issues and eager to take more advantage of primary SPLE solution, add quality to it, and develop it further.

Actions under the level are complimentary to initial adoption activities and contribute to the sustainability and efficiency of the SPLE practices. Defined processes, more attention to quality and communication, and stability characterise this level.

Improving level (level 3) takes advantage of different pre-defined, available and comprehensive methodologies in SPLE paradigm. The level and EAs on it are meant for companies who have achieved the stability of SPLE processes but have to extend them because of the growing production needs.

It can be seen as a step where organisation grows further from medium size and need more rigidness and comprehension in its processes.

4.3 Notation

SPLEMM is presented as hierarchical list of model elements. Every MA, PA, SPA and EA in the model has a unique identifer which denotes its place in the model structure. See section **??** for detailed descriptions of model elements. Elements are presented in the model following way (Figure **??**):

1. - Code
   Unique identifier that each element in the model has. The letter in the code denotes the MA where element belongs to. Numbers indicate element's position in the hierarchy.

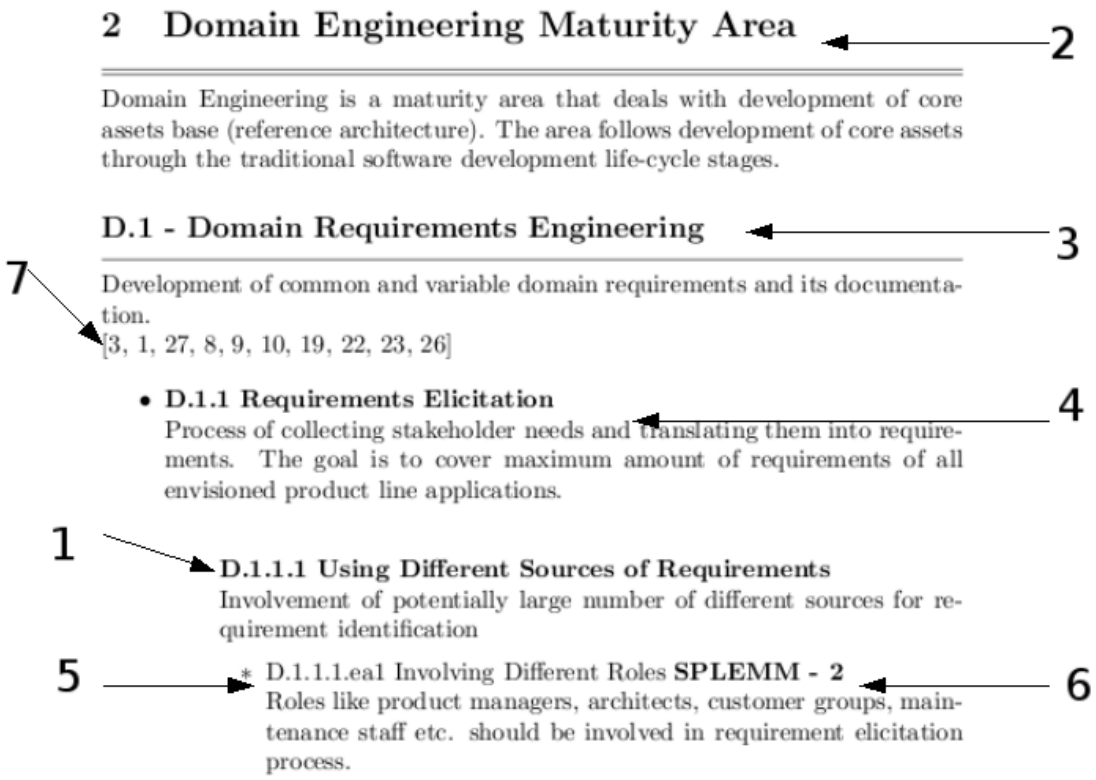

Figure 10: Example of SPLEMM structure

2. - Maturity Area (MA)
   The model consists of four maturity areas that each have its unique distinguisher (B - Business, D - Domain Engineering, A - Application Engineering, C - Collaboration).

3. - Process Area (PA)
   Highest element in the model after MA.

4. - Sub-Process Area (SPA)
   Elements that may be in several leveles between PAs and EAs.

5. - Example Activity (EA)
   Bottom elements of each branch in SPLEMM.

6. - Maturity Level
   Denotes the level of maturity for EAs.

7. - References
   Each PA has list of references attached to it. This presents the main sources that were used to collect and analyse the information under under this branch of the model.

4.4 Evolution of the model

FEF [41] as the most sophisticated maturity model was used as a main reference for SPLEMM in the beginning of the project. Inspired by that the BAPO structure was tried tested. By BAPO the model was divided into business, architecture, processes and organisation areas (Figure 4). Soon the need for better expression of two-life-cycles of SPLE (Figure 1) was recognised.

Reason for that was that two-life-cycle approach is one of the fundamental principles (2.2.2) of SPLE and the model was intended to give an overview of the SPLE paradigm (**??**). Thus it was assumed that the model gives better understanding about SPLE when two-life-cycle principle is presented already through the high level structure. Also some difficulties were experienced when the BAPO model was extended with practice areas which often apply to either domain or application engineering. Thus the original architecture maturity area was divided into domain and application engineering areas taking Pohl et al.' book [31] as a model. Similar categorisation is mentioned also in SEI Framework for Software Product Line Practice (Figure 5). Also the processes and organisation areas from BAPO were strongly changed as they were considered too abstract and unsuitable for model' purposes. These areas were combined into one *Collaboration* maturity area.

Final structure evolved during theoretical study where maturity areas were populated with sub processes. Structures of several existing models [38, 21, 41, 9] were analysed to find suitable combination to best fulfill the research goals (**??**). Lot of features were omitted in order to keep the model simple and easily understandable. Hierarchical structure, concept of success factors, satisfied-explained, mandatory and optional elements, and maturity levels were developed or obtained during the studies.

Number of modifications in the content of the model were also done after analyse of validation results. These changes are listed in chapter 5.4. Final version of the model is available as appendix of current document.

4.5 Usage of the model

The model was created to make SPLE paradigm more accessible for SMEs. For this purpose model's functionality had to cover guidance for SPLE adoption, assessing and benchmarking current SPLE efforts, and also an overview of SPLE paradigm and estimation of the level of changes necessary for adoption.

4.5.1 The model as a source of SPLE overview

It is intended that the structure of the model gives its users an understanding of process areas that have a role in successful SPLE practice. This is a strong addition to the definition of the model by giving the overview of SPLE's scope. Example actions allow users to get initial understanding of the level of changes and complexity of actions needed in their organisation for SPLE implementation. Simple reading of the model and comparing the model to the processes in organisation can already give sufficient initial overview of SPLE and related actions. Gaps between the model and practiced processes pinpoint additional changes and extensions that organisation must execute in order to adopt SPLE paradigm.

4.5.2 First time adoption of SPLE

Organisation that decides to adopt SPLE practice using SPLEMM should first map its existing practices to the model - do a process assessment. This will give them an initial understanding of their current state and rate of necessary changes in their processes. This kind of mapping is easier when organisation already is following some process framework and has an structured view of their processes.

Organisation should take a top-down approach on process assessment. This means that it should go through elements in each branch of the model starting with maturity areas. On each process area it should be checked if it is performed in organisation. Notes should be made on whether element is done or not done. Also any variations from elements definitions or other comments should be stored. Extra attention must be paid on procedures that are considered as a success factors in company's existing processes but are not in the model. These should be considered as a parts that should be added to the model (see section **??**).

For example organisation may discover that under Domain Requirements Engineering PA (D.1 - see table **??**) they have *Domain Requirements Elicitation* SPA (D.1.1) and *Requirements Management* SPA (D.1.3) performed but they are not doing *Commonality and Variability Analysis* (D.1.2). They may also notice that besides EAs listed under *Using Different Sources of Requirements* SPA (D.1.1.1) they have some additional sources that they plan to keep using when adapting SPLE. These will be added to the model as additional EAs.

This assessment of *Domain Requirements Engineering* PA showed that organisation should extend their current processes with whole *Commonality and Variability Analysis* SPA (D.1.2) to successfully implement SPLE in organisation. Analyse of the example activities on the lowest level of *Commonality and Variability Analysis* branch allow organisation to predict the amount of work needed to implement the changes.

Code	Title	Maturity level
D.1.1	**Requirements elicitation**	
D.1.1.1	**Using different sources of requirements**	
D.1.1.1.ea1	Involving Different Roles	2
D.1.1.1.ea1	Involving different sources	2
D.1.1.1.ea3	Studying existing applications for requirements	2
D.1.1.2	**Initial organisation of requirements**	
D.1.1.2.ea1	Prioritisation of Requirements	1
D.1.1.2.ea2	Separation of Problem and Solution Space	1
D.1.1.2.ea3	Defining Requirements Organisation Process	2
D.1.1.2.3a4	Using Proven Representation Methods	3
D.1.1.2.ea5	Categorise Requirements	2
D.1.2	**Commonality and Variability Analysis**	
D.1.2.1	**Identify a Set of Common and Variable Requirements**	
D.1.2.1.ea1	Application-Requirements Matrix	1
D.1.2.1.ea2	Priority-Based Analysis Scheme	2
D.1.2.1.ea3	Check-List Based Analysis	2
D.1.2.2	**Requirements Variability Documentation**	
D.1.2.2.ea1	Defining Variation Points and Variants in Requirements	1
D.1.2.2.ea2	Distinguish between internal and external variability	1
D.1.2.2.ea3	Document the common requirements in detail	2
D.1.3	**Requirements Management**	
D.1.3.1	**Manage Requirement Changes**	
D.1.3.1.ea1	Communicate Software Requirements	1
D.1.3.1.ea2	Process for Integrating New Requirements	2
D.1.3.1.ea3	Collecting Feedback From Application Engineering Cycles	2
D.1.3.1.ea4	Reactive Customisation of Product Line Requirements	1
D.1.3.2	**Guarantee Quality of Requirements**	
D.1.3.1.ea1	People Carrying Knowledge	
D.1.3.2.ea2	Document Assumptions of Commonalities	
D.1.3.2.ea3	Compromise Between Quality and Generality	2
D.1.3.2.ea4	Regular Roundtable Meetings	2
D.1.3.2.ea5	Usage of Defined Requirement Management Tools or Processes	3

Table 1: Structure of Domain Requirements Engineering PA

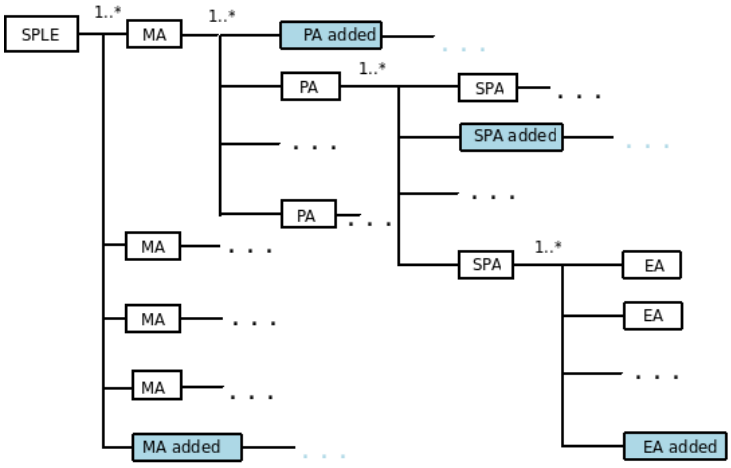

Figure 11: Extending the model on different hierarchy levels

When missing parts of the model are identified it has to be found out whether they should be adapted fully or should some parts (or whole area) be considered as *satisfied-explained* (see section **??**) and excluded from the organisation SPLE processes. This decision should not be taken easily and leaving out an element from the model requires thorough analyse and motivation by SPLE expert. Thus it is suggested at least on 1st maturity level not to leave out anything from the model without having very strong and obvious reasons for it.

4.5.3 SPLE process assessment and improvement

Maturity levels give a basic opportunity to first identify the level of organisation's SPLE processes and also to compare the SPLE levels of different organisations or different units working separately in the same organisation.

Mapping current processes to related SPAs in the model - doing process assessment (see section **??**) - allows comparison of existing actions with the example actions in the model or the ones performed in other organisation under same SPA. Detailed categorisation of the processes simplify the improvement attempts. For example when there are two units in an organisation that follow individual non-standardised SPLE processes then both of the processes should be assessed. By comparing the activities performed and deviations from SPLEMM it is easyer to identify which practices lead to success in certain process areas and standardise those in all units of organisation.

4.5.4 Modifications on SPLEMM

For the model to be tailorable and up to date in future it has to be possible to make changes in it without altering the general principles and structure of it. Model could be changed when new information evolves from research or when practitioners want to tailor it for use in smaller and more specific target groups (eg. specific industry or single organisation).

The developed structure of SPLEMM allows it to extend the model with new information. Users of the model can add elements to each level of the

model under the root element SPLE (Figure **??**). Changes can be made among the levels of EAs (example activities), SPAs (sub-process areas), PAs (process areas) and also on MAs (maturity areas) (Figure **??**). The higher is the level where change is made the more careful has the user be when performing it. When EAs are optional elements that are meant to be modified and tailored, then changes on all the mandatory element levels have to be carefully considered and motivated. For example It is likely that some organisation needs to add new SPA under *Budgeting and Investment* PA (B.3.2) to accent some company specific budgeting system. First they should control if it would be possible to add the element as EA and then think about adding it to a higher level. All the initiatives of adding new MA or PA should be considered extra carefully as the effect on the model's behaviour is much higher on these levels of abstraction.

When adding an element to any mandatory level of the model sub-elements have to be provided as well. The success factors of the added element have to be found and described in the model for it to be complete.

It is also possible to remove and modify the elements of the model when applying the same rules. The changes on the higher level should be considered as more rare and proper motivation is needed before implementation. Also it has to be controlled that EAs are on lowest levels of all the branches.

4.6 Limitations of the model

Constructed SPLEMM consists of software product engineering process parts that are related to SPLE paradigm. Thus it does not cover all the areas of SWD processes. This compromise had to be done to keep the model from growing too big. It may be beneficial to use SPLEMM together with more general models to gain improvements in areas outside of direct SPLE scope.

It has been hard to avoid relative abstraction of maturity levels and achieve the planned purpose of them at same time. Thus the maturity levels leave some room for interpretations. Due to lack of theoretical and empirical materials in many areas of SPLE, there is currently lot of room for the model to expand and evolve in near future. Thus the maturity levels were defined with high abstraction level that was found suitable for the early stage of evolution.

It was tried to use only proven theories of success factors to fill the model. Sometimes the lack of case studies about specific topics forced to make compromises in model's contents. Some areas, that could not be tied with concrete usage examples, were left on the optional level of EAs. Thus some EAs in the model are not atomic, concrete actions. In future, when suitable case studies emerge to prove their reliability, these should be changed into SPA status and extended with EAs.

These areas are usually suggested by some theoretical works but by author's opinion were not sufficiently proven in practice to include it as mandatory part in the model. Often sub-activities that could be potential EAs in future are given in descriptions of such SPAs. For example EA *Organising Variability on Various Levels of Product Line* (C.1.2.ea1) was included as EA to the model although it has sub elements of its own. But as no real life usage example was found with this theory it was left as EA with potential to be developed into SPA in future.

Part of meeting	Time
Intro to SPLE	3 min
Introduction of SPLEMM	3 min
Company overview by interview subject	5-10 min
Questions about Business maturity area	20-30 min
Questions about Domain Engineering maturity area	20-30 min
Questions about Application Engineering maturity area	10-15 min
Questions about Collaboration maturity area	10-15 min
General questions	10-20 min

Table 2: Agenda of conducted interviews

5 Validation of the model

Purpose of this section is to validate the accordance of the model to the requirements specified in the research question **??**. The project aimed to create a SPLE maturity model that would be usable by small and medium sized organisations. Series of validation interviews were conducted with organisations from the model's target audience. These interviews, analyse of results and conclusions drawn are described in the rest of the current paragraph.

5.1 Composition of interviews

Validation interviews (2) presented the SPLE paradigm and the developed SPLEMM with the aim to collect feedback from industry.

Prior to the interview just a brief introduction to SPLE and SPLEMM was sent to interviewees. Besides that main concepts were repeated in the beginning of each validation meeting to give attendees an idea of the context. Main part of the interview went through the model and for each SPA it was asked if that or something similar is performed in the company.

Questions were asked in a form "Is this SPA performed in your company?". For example the question about SPA D.1.2.2 was formed as "Is Requirements Variability Documentation performed in your company". Limiting the questions only with a title of a SPA gave additional feedback about the clarity of namings in the model. Interviewees' understanding about the SPA was controlled and guiding definition of SPA was given if necessary. SPA was marked as needing a better wording in cases when interview subject had problems understanding the SPA after initial question.

If SPA was not performed in the company then reason for that was asked. This information helped to analyze the relevance of each SPA later. Non-performed SPA was potentially non-relevant to the organisation. Later analyse had to answer if the SPA is irrelevant only in the concrete case or could the assumption of irrelevance be extended to the whole target group of the model. In latter case removal or modification of the SPA was considered. If SPA was performed then most important actions under that, and factors for its success were enquired. This information helped to extend the model with new EAs (Example Actions) and also to test the relevance of example activities collected from theoretical study. About every PA it was asked if something is missing in the model under this area. Any comments given to this question were analysed.

Modifications were made to the model if the comment were found to apply to whole target group.

Several general questions were asked in the end of the meeting to get final overview about interview subjects' understanding about the model and its functionality. These questions were following:

- Is something missing from the model?
- Key differences between smaller and bigger SWD organisations?
- How well do you understand SPLE paradigm (in a scale 1..5)?
- Why don't you use SPLE in your company?
- How suitable is the model's structure for SMEs?
- How suitable is the model's contents for SMEs?

5.2 Interview subjects

It was intended to conduct validation interviews with organisations that could be potential users of SPLEMM-like model. For that purpose small and medium sized companies were approached. Another interviewee selection criterion was organisation's potential for SPLE. Company had to be a user of SPLE practices or have a potential for that paradigm. Thus ideally, in case of further interest from company side, they could use the model for process assessment and improvement activities or use the model as a tool for SPLE adoption. Interview subjects with given profile were chosen to make it simpler for them to relate with the model and with introduced SPLE paradigm. Also these prerequisites set to interview subjects made their provided answers more relevant in terms of research question. Interview subjects' correspondence to criteria was controlled through initial observation of available information about the subject and phone-discussion with company representative. Organisations were expected to have experience with development of one or preferably more products in defined domain. No official potential analyse like PLPA (Product Line Potential Analyse [20]) was considered necessary. In ideal case the final model should be simple and intuitive enough to be suitable as first choice tool for both initial screening of SPLE potential and further adoption activities. Thus it was intentional to have organisations with uncertain SPLE potential as interview subjects. These characteristics of interviewed organisations made it possible to evaluate suitability of SPLEMM for getting an initial overview about the paradigm.

Four organisations agreed to participate in validation interviews (Table 3). Syntronic is an international engineering design company with a smaller office in Goteborg. The organisations is specialised in the design and development of electronics, electro-mechanics, technical and administrative software in telecommunication, automotive, defense and medicine industries.[7] Interview was made with the head of their Goteborg's office that was looked as an isolated part of rest of the corporation to study the questions in SME similar environment.

Carmenta offers a wide range of software products for business-critical geospatial applications with main markets on location based services, security and defense, and aerospace. [1]

Organisation	Nr of employees	SPLE practitioner	Products and domain
Carmenta	approx. 60	no	software products for geospatial applications
Lavasoft	approx. 60	no	several product versions of spyware protection
Tibco Spotfire division	approx. 150	no	information visualisation and analytic tools
Syntronic	approx. 250	no	consultants with wide experience in product development

Table 3: Interviewed organisations

Lavasoft is a creator of widely used spyware protection software Ad-Aware. Their product selection consists of several configurations of the base software that can include several levels of additions to basic protection. [3]

Tibco Spotfire division provides users with information visualization and analytic solutions tools [6]. The company was recently bought by Tibco corporation but was able to remain relatively independent in the questions concerning areas of SPLE.

Each organisation was asked to provide an interviewee or interviewees with experience in organisation's software development and product management processes. Interviewed persons held positions of office manager, project manager, product manager and R&D manager. 1 interview with Regional office manager was conducted in Syntronic and 1 interview with project manager in Lavasoft. Product manager was interviewed in Spotfire.

In Carmenta R&D manager and two project managers were interviewed. Introductory part to SPLE, the model and the company was performed with all attendants present. Rest of the interview was divided into two parts where questions about the Business maturity area were asked from product managers and questions about Domain and Application Engineering maturity areas were asked from R&D manager. Questions about Collaboration maturity area was asked from both roles. As R&D manager was attending through whole interview process he was subject to general questions in the end of the meeting as well.

Each of the interviews were conducted in subject's facilities in order to cause minimal disturbance to their daily work. Notes were taken both on laptop and paper. Guiding materials in form of process descriptions and diagrams were received from interview subjects in some cases to clarify more complicated processes and methods. These documents are not used directly in the current work but used as references together with interview protocols.

Interview protocol and results were summarised and analysed after each meeting to be able to store the fresh impressions from the interviews. Further analyse of the information took place when collected information was ready for comparison. It was agreed with the interviewees that any further question would be clarified via phone.

5.3 Results and analyses

Conducted interviews gave feedback to the developed model. Interview protocols consisted of comments to separate process areas as well as to the model in

general. After the interviews each process area was reviewed taking into consideration the information collected. None of the interviewed organisations were knowingly using SPLE practices. Due to that many SPAs that were more SPLE specific were commented as "non performed" in interviewed organisations.

General feedback to the model was positive. Interviewees admitted the improvement of their knowledge about SPLE and its implementation issues after going through the maturity model during validation meetings. Design of the model's structure can also be considered successful and easy to follow for users.

Most of the interviews' time was spent on a discussions about Business and Domain Engineering maturity areas. Collaboration and Application Engineering maturity areas were gone through faster. Organisations did not practice SPLE and due to that the processes under these maturity areas were not recognised in the organisations.

5.3.1 Feedback to the structure and content of the model

In order to meet the usability and usefulness requirement of the model (see section **??**) SPLEMM development initiative aimed to create a model with logical and easily understandable structure. Developed model consists of process areas that are organised into hierarchical tree-structure. SPLE practices are organised into 4 maturity areas that each have hierarchies of sub-elements that contribute to the success of the area. Example actions are given on the lowest level of the model. In order to keep the model from growing too big only process areas that are directly influenced by or contributing to SPLE practice are included. Some process areas that are part of general SWD frameworks (see section 2.6), but not so much related to SPLE adoption in an organisation, are omitted from the model. For example project management issues are separate process category in CMMI [38] but in SPLEMM they are not included as they are not directly affected by SPLE characteristics.

Structure of the model was presented in the beginning of the validation interviews and followed throughout the meetings. In the end of the meetings interview subjects were asked about their opinion about suitability of the structure for the model's purposes. Also any kind of feedback they had regarding the structure or missing parts of the model was recorded.

General opinion about the developed structure of the model was positive. It was accepted as suitable for the purposes and target group. The structure was regarded as easy to understand and follow. The 4th maturity area, Collaboration, raised some questions. When Business, Domain and Application Engineering were quite distinctive areas then Collaboration seemed to be the MA with more abstract definition. Concrete proposals were made to consider dividing Collaboration process areas between other maturity areas or name it as *Miscellaneous* or *Technology Management* MA.

Feedback about the model was collected throughout the interviews. Each SPA was presented to the interview subject who then had an opportunity to comment the relevance of it among the organisation's processes. Not much additions were proposed during the interviews. The model's content was considered sufficient taking into account that it was not an aim to cover all aspects of software product development but only the areas that are tightly related to SPLE. When interviewee recognised a SPA as being practiced in an organisations he was asked to name concrete actions that are performed to guarantee the suc-

cess of the practice area. These inputs were used to specify and add example activities to the model.

Following four subsections describe interview findings in four maturity areas.

5.3.2 Validation of the Business maturity area

This subsection points out the general trends and relevant findings that came out during the validation interviews about Business maturity area.

Some form of Marketing strategy existed in all of the organisations. Promotion and development of brand name was considered important and was mainly exercised through the usage of common brand name throughout all the products. Although exact usage of brand was different in the organisations, then importance of this asset was recognised in all cases. One organisation had recently renamed its products to unify the naming of product and organisation. Other company had strong enough brand that even after the company was bought by another corporation its products remained on market with the old name.

Relationship Management (D.1.1.3) was based on a quite similar principles throughout the organisations and actions pointed out as performed during the interviews were matching with example actions in the model. *Market strategy* (D.1.1.2) had more varying approaches among the companies. Main differences were due the level of detail used for defining the market strategy and its generation processes.

Scoping processes in the studied organisations were quite straightforward and lightweight. Only one organisation had distinguishable procedure defined for it and incorporated into organisation wide product management processes. Criteria for domain and scope selection was not defined but decisions were made on product managers meetings. Scope revisions were usually informal and it was up to project managers to get the new information from different sources and regularly discuss the scope.

Business vision and strategic goals had a high relevance in all the organisations. Goals are set, followed and tracked for several years terms. Although not as much as in SPLE case studies, the relationships between strategy and production characteristics was recognisable in interviewed organisations.

Financial management practices existed on different levels of rigidity and comprehension in the organisations. General trend was that budgeting and money investments were made on high-levels and did not go very much into details. Organisations where core asset development was more distinguishable could the tracks of it be seen also in financial management. In given cases decisions on new products were not motivated with payback period, net present value, internal rate of return or other similar techniques [8].

Summarising notes:

Lessons learned - Interviewed organisations were familiar with goals and principles introduced through *Business* MA process areas. *Scoping* was an area where processes described in SPLEMM were distinctively more comprehensive than the ones actually used in the organisations. Common trend was to use light-weight and in-formal methods throughout whole maturity area. Still the number of processes from the model that was recognised by interview subjects was surprising.

Before the validation it was doupted if separation of strategic goals and business vision under separate SPAs in *Strategic Planning* SPA (B.3.1) was well-founded. Interviews proved that the organisations distinguish between these areas and they can be kept apart in the model.

Validation process - In some cases interviewed persons did not have a full overview of marketing and financial management activities. They could provide a general overview about these areas in company but no concrete details. In current case this did not affect the outcome of the result as the information was sufficient for the model validatoin task. Still the wide scope of SPLE should be considered in future works.

Impact on the model - Based on the validation meetings it can be said that success factors under Business maturity area are relevant also in the interviewed organisations. This information contributes positively to fulfillment of relevance requirement given in chapter **??**. As scoping practices in organisations were not following all the SPA-s presented by SPLEMM some simplifying changes were considered in the Scoping PA. For that related theory was analysed again. It was recognised that the importance of scoping is much higher in SPLE environment. thus it was decided that reasoning given during validation was not sufficient and *Scoping* PA was not modified. Contrary example was with *Strategic Planning* (B.3.1) where SPAs concerned with strategic goals (B.3.1.1) and vision (B.3.1.2) were proven to be distinguishable. Thus these both were remained in the model although it was considered to combine them before the interviews. Some changes were still made in the Business MA. See section 5.4 for a full list of those.

5.3.3 Validation of the Domain Engineering maturity area

Interviewed organisations did not have comprehensive or rigidly defined requirements engineering processes. Still the importance of existence and quality of the process area was commonly appreciated. Amount of requirement sources was wide including for example customer feedback, competitor product screenings, input from partners and changing trends in related technologies. Organisation of requirements was light-weight and had some minor variations between the companies. Still some form of prioritisation was mentioned by every interview subject. Variability in requirements was not understood in the way as it is described in SPLE paradigm. General requirements applying to all products and products specific requirements are distinguished. But variability in individual requirement items was not used. As stated before, requirements were regarded as highly important artifacts in the organisations and need for improvements in the quality was mentioned in validation interviews. Organisations had used different solutions for improvement of requirements engineering processes previously. Agile principles were used to remove communication gaps between requirements engineers and software developers. Noteworthy was an example where company had reassigned the task of requirements writing from team based in USA to Sweden in order to have requirements engineers and developers working in same teams.

Organisations were using rather light-weight solutions also in *Architecture Definition* PA. Agile practices like open communication, short iterations and

light-weight documentation [15] were enforced. All the organisations were mentioning modularisation as a goal for their architectural strategy and this affected decisions throughout the whole life-cycle of software development. Interviewed organisations were often using 3rd party components and software developed by suppliers. Thus *Buy/Make/Mine/Commission Analyses* SPA (D.2.3.3) had high relevance for them. Besides the concern if externally developed component could break their modularisation efforts, the need for keeping strategical knowledge in-house was regarded as important factor under this SPA. Methods for ensuring stability of architectural style were dedicated board of architects and usage of in-house SDK (Software Development Kit). The board of architects contributed into quality of architecture by coaching new developers and validation during code reviews. It was also considered as a good practice to assign each architect for a specific part of product's architecture. Maximising the use of in-house SDK was also acknowledged as a way to keep constant architectural design.

Interface and component designing processes were distinguishable in the organisations. Interface design was considered extra important by the organisations that managed their own SDK (3 out of 4). In these cases interfaces were kept well documented and up to date with new technology trends. Also the aim for clean and simple architecture was confirmed under this point again. Common design practices like object oriented programming, test driven methodologies and maximising the use of existing systems were used to guarantee the quality in interface and component design stages. Compilation process was automated as much as possible, for example through usage of continuous builds and nightly tests. It was pointed out during one interview that considerable troubles are raised by 3rd party components that are not applicable with automated build processes.

There were several ways used for architecture evaluation. One organisation had set peer review requirement for all the code that is committed to repository. Others had more relaxed view of code reviews and held it in every 1 or 2 weeks for teams of developers. When two organisations had dedicated quality assurance teams for testing then others had assigned quality assurance tasks to developers. Still even then it was stressed that different people have to develop and test one code. 2 out of 3 organisations brought up that usability testing requires quite a lot of resources.

Documentation was considered to be rather unprioritised area. When customer documentation was always kept up to date then technical documentation was often found in a bad shape. Importance of technical documentation was not considered very high. For example new people were introduced to the work rather through communication and mentoring than with technical guides. Technical documentation was often maintained on a level of code only. Still there was an exception in an organisation who had built up an full internal process for technical documentation.

Generally it can be said that process areas in *Domain Engineering* MA are relevant to interviewed organisations and cover the their software development processes sufficiently. As only few process areas were not recognised it can be concluded that adaption effort here would not be too high.

Summarising notes:

Lessons learned - Main thing that was confirmed with the interviews was that due to SMEs' tendency to practice agile methodologies, their processes are more light-weight than the ones described in SPLEMM. However it was noted that none of the organisations had fully implemented any concrete agile methodology. Rather general principles were used and different practices tested now and then in a search for the best suitable method. So low level of documentation and process comprehension was not an aim itself but a method that was belived to help in creating better software. It was agreed that if SPLE can fulfill that goal then there should not be principal opposition against the paradigm.

Validation process - *Domain Engineering MA* got very good coverage during the validation interviews because the interview subjects understood it best as it follows the general flow of traditional software development process. The MA took most time during the interviews and also most information was collected there. Thus it might seem that domain engineering has also so much higher importance in organisations' processes. Actually it must be understood that interview subjects were often refering to whole software deveopment area in organisation when answering questions under *Domain Engineering* MA part. Interviewed organisations were producing products as single systems.

Impact on the model - Differences between light-weight processes of SMEs and SPLE that needs some more comprehension are not too big. Besides variability related PAs the adaption of other SPLE domain engineering activities should not require much change in activities.

Interviews in this MA validated number of EAs that were mentioned in some theories but not documented as tested in practice. These were the EAs under *Domain Requirements Engineering* PA (D.1) and *Architecture Definition* PA (D.2).

Testing processes in organisations were not much different from common single system testing procedures. Even in cases where some work was done on core assets in R&D level, the main part of testing was performed on applications. So the information about testing was not included to the model as nothing SPLE specific was found. The same issues were experienced with documentation, as no addition to the model was found during the interviews.

5.3.4 Validation of the Application Engineering maturity area

As mentioned before the interviewed companies were not users of SPLE paradigm nor did they have separated life-cycles for domain and application engineering. Instead they had singe process for product development. Thus lot of the information about Application Engineering, that was related to general software development, was collected through questions about Domain Engineering maturity area. The part of interviews about Application Engineering maturity area tended to take less time and were aimed to give hints about additional similarities or differences between SPLE practices and practices in an interviewed organisations.

The organisations had main requirements engineering practices related to applications. This means that application specific requirements are collected and analysed. General requirements, that are found to be applying for all products, are identified during analyse phase. So practically everything mentioned in last subsections holds also true under Application Engineering.

Traceability in artifacts was not enforced much. Again organisations relied on close communication instead of more formal tools and processes. In one organisation commonly used issue and configuration management tool allowed to implement traceability from requirements until parts of code that realise them. But even there it could not be answered by interviewee how big is the benefit of usage of such system.

Some high level configuration of variants could also be recognised. Most of the organisations (3 out of 4) were offering variants of their core product to customers. Products differed on a set of included features. These variants were binded and configured in compile time. With simple configuration file triggers it was possible to include and exclude the features of projects. Still physically all the code was included into the product. Just certain features were disabled on lower levels of products.

General software development processes in interviewed organisations were not very distinctive from the processes described by SPLEMM. Although there do not exist 2 separate life-cycles, most of individual process areas are known to the organisations. Even the concepts of variability binding (A.2.3) and asset mining (A.2.2.1) were not totally new to these companies.

Summarising notes:

Lessons learned - Interviewed organisations already have the concept of deriving different product versions out of one code base. But this is far from the level of organisational and technical coordination of SPLE paradigm. It can rather be described as a part of compilation and build activities of software development than anything related to two-life-cycles model of SPLE.

Validation process - It was hard to get information about *Application Engineering* MA during the interviews. As the interviewed organisations were not practitioners of SPLE they had not implemented the two-life-cycle model. Thus the validation done in the area can be considered somewhat incomplete.

Impact on the model - Validation of *Application Engineering* MA proved that division of core-asset development and application development into separate MAs (see section **??**) was beneficial. Single-system developers need thorough presentation of two-life-cycle model concept and other SPLE fundamental concepts (see 2.2.2).

Besides adding one EA (*Using Subject Matter Experts* - A.1.1.1.ea2) no more changes was done in the model. Still it was felt that area of application engineering in the model needs validation with organisation who have actual experience with SPLE and application deriviation processes.

5.3.5 Validation of the Collaboration maturity area

Not much was done in the field of *Variability Management* (C.1) in studied organisations. The area was existing in form of keeping track of general requirements and allowing some high level changes through configuration on compile time. Still it can not be said that any of the organisations had a planned approach to the PA as described in SPLEMM.

Automation seems to be the main followed principle on configuration management. Noteworthy practices here are automation of compile and integration processes, continuous integration processes, and dedicated and skillful configuration management responsible. The processes seemed to be straightforward for the organisations. Only Spotfire mentioned some issues that raised with the purchase of another organisation. In that case it had been hard to unify the two already working systems. Outcome of these efforts were not clarified by the time of writing this document.

Among the organisations it was common that only one team was dealing with all produced products. Its tasks could be compared to the ones of application engineering teams' in SPLE paradigm. As a counterpart for SPLE's domain engineering team two interviewed organisations had R&D team which were developing core set of features of product lines. Even as their goal was not to create specific reference architecture, this kind of structure could be compared to SPLE organisation and could serve as a catalyst for cultural change.

Organisations have been trying out different process standardisations and SPI initiatives. These vary from ISO standards to CMMI and agile practice adoptions. General trend seems to be the use of trial and error method where failures are accepted and fast feedback expected. Dedicated PM, supporting technical lead and open communication were mentioned as most common success factors for the SPI processes in these organisations.

Summarising notes:

Lessons learned - Most of the PAs in Collaboration MA were not unknown to interviewed organisations and many common aspects were found that connects them to SPLE practices. Interesting observation was that some organisations already had something comparable to core asset team. Their R&D teams were developing features that would be used in all the product line products. 3 out of 4 companies had had experiences with some process standards. This experience potentially improves organisation's ability to adapt SPLE [31].

Validation process - Again as the interviewed organisations did not have experience with SPLE it was not possible to objectively validate Variability Management PA. Only one of the interview subjects had had direct experience with SPI project in an organisation but this was not really a problem as other interviewees were able to give sufficient information about these initiatives.

Impact on the model - Not many changes were initiated after validation interviews in the *Collaboration* MA. As the organisations were not fully using the PAs it was not possible to run a full validation on those. Also nothing spectacularly SME or SPLE specific were found about PA C.3

(*Organisation*). The general findings in the area was usually not suitable for the model. Still one EA was added under *Roles and Responsibilities* PA.

During the theoretical study PA C.4 was planned to be added to the model to cover the issues related to SPLE adaption project. Although some information was gained during the validation that impoved the overview about SPLE adaptability in SMEs, it was still found to be not sufficient to support whole process area. The PA could not have added true value to the model and thus was decided to omit.

5.3.6 General observations

Although the organisations were mainly product oriented and were not familiar with SPLE practices before, some hints about two life-cycle way of thinking could be recognised during the interviews. Organisations had very specifically defined their core competencies based on what they had built the core software on what product laid on. For example Lafasoft had Ad-aware core product which was sold to different customer segments by adding several features to it that are built in house or by 3rd parties. In their case also existed a team that was dedicated on a work with core system development.

Based on interviewed organisations it can be said that many SPLE principles are already used in software product developing SMEs. This has two contrasting effects on SPLE appeal to them. First it lowers the adaption barrier and effort needed for implementation of the paradigm. Instead of complete change of processes some practices has to be modified and some can be left as they are. As interviews showed the recognition of SPLE practices in organisations that consider the adoption can also have a negative effect on adaption. Interview subjects were asked about the reasons for not applying the SPLE in their organisation. One reason was that when seeing big part of the SPLE already done in the organisation then current level seems sufficient and further changes and investments are perceived with doubt. This stresses the requirement of the model for giving a fast overview of adoption effort and achievable benefits. The problem can be solved when stressing the benefits of full solution when introducing the SPLE paradigm.

Also there were process areas found that are in SPLEMM but were not performed as part of processes in interviewed organisations. One of the reasons was that interviewed organisations were not users of full SPLE paradigm. Thus they are not using many SPLE specific practice areas. Other group of SPAs that were recognised as non-performed can be considered as not relevant in SME settings. This distinction was clarified with brief discussion during the interview and following analyse of the SPAs. For example none of the interview subjects recognised *Requirements Variability Documentation* SPA (D.1.2.2) because it is very specific to SPLE practice. Number of other variability related SPAs were not performed by interviewed organisations. Contrasting example is *Understanding Product Line Architecture Requirements* SPA (D.2.1.1), that has a low priority for SMEs. This SPA is important to unify the understanding of requirements between people who document those and people who use the requirements for development. When in big corporations this is an actual problem because different people are involved in those activities then in SME environments requirements are documented and used by same teams.

Interviewees were also asked how do they see the differences between SMEs and big SWD organisations. The most praised characteristics of SMEs were flexibility and agility. Lower number of people, open communication and less rigid processes were proposed as contributors to lower overhead. Obvious disadvantage named in one of the interviews was "lack of muscle" in SMEs. The amount of resources sets some limits to possibilities of development speed and reach. With limited team sizes it is crucial in smaller organisations to select right people. People has to be selected more carefully to suit professionally for the tasks and personally to the teams.

5.4 Changes in the model

Not all improvement ideas suggested or found during the interviews was transferred to the model. Each change suggestion that was found during the interviews was analysed to find out if the change would apply with SPLEMM criteria and requirements (see chapter **??**). For example one organisation suggested to add example activity under *Brand Name Strategy Aligned with SPLE* SPA (B.1.1) that suggests keeping the company name visible on product names. This change was still omitted as it is rather general branding strategy issue and not related to SPLE. Also it was needed to be careful when discarding elements from the model. Thorough analyse had to be done to clarify if this suggestions is made due to organisation specific issues or due to something that can be applied to the whole target group of SPLEMM.

Following is a list of changes made in SPLEMM after validation of the model:

- **Marketing Strategy PA was renamed to Sales and Marketing** - During the validation interviews it was pointed out that the process area actually involved more than it would be expected from area named Marketing Strategy. It was referred back to Family Evaluation Framework [41] and found that in that model exists an aspect that is named *Sales, marketing, product management involvement*. Analyse of the content in Marketing Strategy PA suggested that renaming is well-founded. Also earlier sub-process area Product Line View to Marketing was removed and its content was attached to the definition of Sales and Marketing PA. Main reason of this change was to increase the simplicity and make it easyer to understand the model

- **Active Involvement of Customers EA (B.1.3.ea5) added** - Two interview subjects showed good results achieved from working with customer groups and bringing different customer groups together. For example organised conferences increased the interaction between customers on different levels and this has a good effect on product development and also on sales. As this activity has a high intercorrelation with several other SPLE activities, the EA was added to the model.

- **Getting Information from Stakeholders (B.2.1.1.ea2)** - The definition of this EA under Scoping SPA was extended to also mention the customers and competitors as possible sources of information. These roles had been missed during initial theoretical study but were stressed in every validation interview. It was decided to distinctively mention customers and competitors under explanation of EA B.2.1.1.ea2.

- **Understand Product Line Architecture Requirements PA was removed** - Validation interviews suggested that this PA was not relevant for organisations in smaller sizes. The aim of this SPA was to eliminate the problems caused by the dendency that in bigger organisations are requirements engineering and actual usage of requirements often done by different people. So there is a need for steps to be made to unify the understanding that different groups of people may have about same requirements. In SMEs these two tasks are usually done by same people. So this SPA can be considered as irrelevant for SMEs generally.

- **Categorise Requirements EA (D.1.1.2.ea5) added** - During the interviews one organisation presented their approach to organise requirements based on categories. This had proved to be useful as company was also doing some iterations where only requirements from specific category were approached. It was decided to add this practice as EA to the model.

- **Definition of Requirements Management Process EA (D.1.3.2.ea2) added** - All the interview subjects recognised the importance of requirements engineering processes. Two organisations did not have defined and documented process for requirements management but still admitted the need for improvements in the area. Two other interviewed organisations had and were using documented requirements management process and stated it as a beneficial practice. Taking into account several mentions of documented requirements engineering processes in theory [38, 21], and its importance in SPLE, it was decided to add this EA to the model.

- **Usage of Agile Principles EA (D.2.4.1.ea1) added** - All of the organisations mentioned the effect that some agile practices and principles have had on results of their development processes. Although it was not included as separate EA in the model before it was now decided to do that. Suitability of Agile and SPLE is discussed before (see chapter 2.5) and validation interviews gave examples of its suitability in potential SPLE environments. Even it can be argued that the usage of agile methodologies is not much different in SPLE and singly system environments, the EA was still added to emphasise suitability of the agile theory in SPLE practicing SMEs.

- **Using Subject Matter Experts EA (A.1.1.1.ea2) added** - During theoretical study the role of expert knowledge in application engineering was not seen as important. Validation interview subjects stressed this area more than expected. Based on later analyse it was decided to add the EA to *Application Engineering* MA.

- **Emphasise Skills and Education EA (C.3.2.1.ea2) added** - Interview subjects brought up the importance of employers professional level in SMEs. Based on different sources it can also be understood that SPLE paradigm sets high expectations to its practitioners. To emphasise this issue, the EA was added under *Roles and Responsibilities* PA.

6 Conclusions

Final section of the document presents the learning from the project, reviews the satisfaction of research question, gives suggestions for further research and briefly summarises the whole document.

6.1 Reflections on SPLEMM development project

The project of creating a maturity model for the whole area of SPLE involved a huge scope. Areas covering business, software engineering, organizational structure, and work process are all intricate parts of a complete model. It is impossible to grasp all those aspects in detail during the project with the given resources. Instead of specific studies on each area big picture had to be focused on. The work gave a good understanding about different parts of the SPLE paradigm and interactions between those parts. Also the general understanding and experience of scientific research initiative is something that is gained from the project.

It proved as important to have an opportunity to discuss and elaborate about the model related issues. The fastest progress in the thesis was experienced during and directly after validation interviews and meetings with supervisor. Working in pairs could definitely be suggested for future projects of this kind.

Choice of the suitable structure for the model was not an easy task. It took some time and lot of work to come up with the structure that contributes to all the requirements extracted from research question (see section **??**). The decision about structure and principles of the project can not be done at once. It is not possible to gather information about SMEs and SPLE and then based on that assemble a working model. This process is rather iterative, where final structure evolves through many changes and modifications (see section **??**). Finalising the structure had a great effect on work progress. It was much more effective to gain new knowledge and organise it in a structured way.

Another interesting issue that came up during the project was a choice of criteria on deciding which information to include to SPLEMM. From one side the amount of information was very big due to large scope of the model. Then again it was found that original intention, to only populate the model with information from theories that are proven in practice, was complicated to achieve. There are not enough experiences documented on such a detailed level to create a full maturity model with so strict restrictions. Thus compromises were needed to deal with lack of information sources. It was not possible to have one to one mapping between success factors and information sources where the success factor was proven. Instead connected theories were analysed and author's judgement used in many cases to decide if potential success factor is suitable to the model.

6.2 Satisfaction of research questions

Number of requirements about functionality and characteristics were set on SPLEMM when research questions of the project (chapter **??**) was disassembled. Although concrete verification projects could be conducted to guarantee the fulfillment of these requirements, initial conclusions about model's characteristics may still be done based on validation interviews and general analyse. Current

section controls how the requirements were met with the final SPLEMM.

Functionality of the model :

- **Overview of SPLE** -

  This functionality was tested during the validation interviews. Interview subjects who did not have prior in-depth knowledge about SPLE were introduced with the concept during the meetings and SPLEMM was used as a main tool for this introduction. After the validation meetings interview subjects were asked to grade their understanding about SPLE paradigm in a scale from 1..5. In each case 4 was given as answer. This gives a good base to assume that the model fulfills this requirement by giving sufficient overview of the SPLE paradigm already with brief introduction.

- **SPLE adoption** -

  The model provides sufficient overview about SPLE. Also it provides first time SPLE users with example activities that allow simple and good reference base for development of their own first time SPLE processes. This combination of overview on abstract level and detailed examples was recognised as potentially well suited for new SPLE practitioners.

- **SPLE process improvement** -

  Definitions of maturity levels of SPLEMM allow to assess the existing processes and see the direction of further improvements. Also it is suitable tool for benchmarking. The model can be used for comparison of processes in different organisations or units of one bigger organisation. Results from this kind of comparison may be used for improving the processes in the organisation that falls behind in benchmarking.

Characteristics of the model :

- **Usable** -

  Based on the feedback collected during validation interviews it can be said that the model fulfills the usability criteria. Interview subjects recognised it as simple and understandable enough. Hierarchical structure was easy to follow and example activities were also acclaimed as feature that simplifies the understanding of the model.

- **Relevant** -

  Relevance of the model's content was checked during the validation. Some changes were made to the model to homogenize it with the state of SMEs. When a SPA was not practiced in an organisation it was taken as a hint that the SPA is irrelevant for this concrete organisation. The reasons for not using the SPA by the organisation were analysed to find out if these may apply to the whole target group of the model. Based on the decision changes were made to the model. These changes and their decision criteria are listed in chapter 5.4. Based on validation and few improvements made it can be said that the content of the model is relevant to the software product developing SMEs.

- **Evolvable** -

  SPLEMM has a structure that is easy to understand and follow. The model is fully extendable with example activities from new case studies in research or user-specific analyse. Also it is possible to add SPAs under every parent element in the model. The structure and principles of the model are built in flexible manner and support evolution.

- **Tailorable** -

  Set of maturity levels is the first option where organisations can choose the settings that are most suitable for their initial situation. Also the satisfied-explained feature of SPA-s give a room for personalisation of the model. Example activities allow organisations to populate the model with activities specific only to their individual process habits.

- **Useful** -

  Usefulness as defined in chapter **??** is hard to prove without testing the model in a dedicated and long lasting case study. During the validation interviews some concerns were presented about cost-benefit ratios of process changes generally. Still it was observed that a large part of SPLEMM processes were present in interviewed organisations' current processes. Thus it can be supposed that necessary changes and investments for SPLE adoption are not too high. Knowing the high rate of reported benefits of SPLE projects in industry, it can be assumed that adoption costs are lower than possible gains from SPLE. Still further case studies are needed to strengthen this statement.

Based on the data gathered in scope of the thesis project it can be said that the developed SPLEMM has potential to fulfill all the requirements set in the beginning of the project. Still further verification and measurements of real implementation cases are needed to confirm the realisation of all the criteria.

6.3 Suggestions for further research

The current project could be continued with validation in organisations that are already practicing SPLE and control the structure and content in these contexts. Next step would be to actually verify the model and test it for real life process assessment, process improvement or SPLE adoption projects.

The need for more industrial case studies in SPLE area was recognised during the project. SPLEMM could be extended with further EAs (example activities) that get proven in these case studies. Author's opinion, that is based on overview of theory and discussions with industry representatives, is that SPLE area currently needs mostly practical success stories that could encourage the wider spread of the paradigm.

6.4 Summary

The project aimed to develop SPLEMM (Software Product Line Engineering Maturity Model) that is suitable for usage in small and medium sized organisations. The SPLE paradigm that has proven itself in larger companies have not achieved wide popularity in smaller software product development organisations.

This gap was targeted with the development initiative of SPLEMM. The model was created based on present best practices and frameworks, and adapted into light-weight form for SME-s. The model was later validated in industry through several interviews that gave input for the refinement and evolution of the model.

It was recognised during the project that many of the organisations among the potential SPLE adopters already have quite a strong base of practices in place. Thus the full adoption of the paradigm would not involve too much changes. Developed model aimed to give an easy way of recognising the gaps to fill for SPLE adoption and achieving the benefits of the paradigm in context of SMEs.

Theoretical and empirical analyse showed that the goal of creating a model that could be used for gaining overview, adopting and improving SPLE processes in small and medium sized enterprises was achieved. The resulting model satisfied the requirements of being usable, relevant, evolvable, tailorable and useful for its target group.

References

[1] Carmenta ab webpage. http://www.carmenta.se/, May 2009.

[2] Definition of small and medium sized enterprises on european commission's website. (http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/index_en.htm), May 2009.

[3] Lavasoft webpage. http://www.lavasoft.se/, May 2009.

[4] Software engineering institute website about cmmi. (http://www.sei.cmu.edu/cmmi/index.html), May 2009.

[5] Software engineering institute's website on software product lines. http://www.sei.cmu.edu/productlines/, May 2009.

[6] Spotfire webpage. http://www.spotfire.se/, May 2009.

[7] Syntronic webpage. http://www.syntronic.se/, May 2009.

[8] John Adams and Linda Juleff. *Managerial Economics for Decision Making.* Palgarve Macmillan, 2003.

[9] Faheed Ahmed. *Process Maturity Model for Software Product Line.* PhD thesis, The University of Western Ontario, 2006.

[10] Vander Alves, Tarcísio Câmara, and Carina Alves. Experiences with mobile games product line development at meantime. In *SPLC '08: Proceedings of the 2008 12th International Software Product Line Conference*, pages 287–296, Washington, DC, USA, 2008. IEEE Computer Society.

[11] 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. In *SSR '99: Proceedings of the 1999 symposium on Software reusability*, pages 122–131, New York, NY, USA, 1999. ACM.

[12] Andreas Birk, Gerald Heller, Isabel John, Klaus Schmid, Thomas von der Maen, and Klaus Mller. Product line engineering: The state of the practice. *IEEE Software*, 20(6):52–60, 2003.

[13] Jan Bosch. Product-line architectures in industry: a case study. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 544–554, New York, NY, USA, 1999. ACM.

[14] Jan Bosch. Maturity and evolution in software product lines: Approaches, artefacts and organization. In *Software Product Lines*, volume 2379/2002, pages 247–262. Springer, Berlin / Heidelberg, 2002.

[15] A. Cockburn and J. Highsmith. Agile software development, the people factor. *Computer*, 34(11):131–133, Nov 2001.

[16] M.A. Cusumano. The software factory: a historical interpretation. *Software, IEEE*, 6(2):23–30, Mar 1989.

[17] S. Deelstra, M. Sinnema, J. Nijhuis, and J. Bosch. Cosvam: a technique for assessing software variability in software product families. pages 458–462, Sept. 2004.

[18] Tore Dybå. Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context. *SIGSOFT Softw. Eng. Notes*, 28(5):148–157, 2003.

[19] W.B. Frakes and Kyo Kang. Software reuse research: status and future. *Software Engineering, IEEE Transactions on*, 31(7):529–536, July 2005.

[20] Claudia Fritsch and Ralf Hahn. *Software Product Lines*, chapter Product Line Potential Analysis, pages 228–237. Lecture Notes in Computer Science. Springer, 2004.

[21] Svahnberg M. Gorschek T. and Tejle K. Introduction and application of a lightweight requirements engineering process evaluation method. In *Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)*, pages 101–112, Essen, Germany, 2003.

[22] Christoph Wienands Gunther Lenz. *Practical Software Factories in .NET*. Apress, 2006.

[23] Geir K. Hanssen and Tor E. Fgri. Process fusion: An industrial case study on agile software product line engineering. *Journal of Systems and Software*, 81(6):843 – 854, 2008. Agile Product Line Engineering.

[24] Michael Kircher, Christa Schwanninger, and Iris Groher. Transition to a software product family approach - challenges and best practices. In *Proceedings of the 10th International on Software Product Line Conference: IEEE Computer Society*, 2006.

[25] P. Knauber, D. Muthig, K. Schmid, and T. Wide. Applying product line concepts in small and medium-sized companies. *Software, IEEE*, 17(5):88–95, Sep/Oct 2000.

[26] Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2):131–183, 1992.

[27] F. Navarrete, P. Botella, and X. Franch. How agile cots selection methods are (and can be)? pages 160–167, Aug.-3 Sept. 2005.

[28] Muhammad A. Noor, Rick Rabiser, and Paul Grnbacher. Agile product line planning: A collaborative approach and a case study. *Journal of Systems and Software*, 81(6):868 – 882, 2008. Agile Product Line Engineering.

[29] Jorma Palo Pasi Kuvaja1 and Adriana Bicego. Tapistrya software process improvement approach tailored for small enterprises. *Software Quality Journal*, 8(2):149–156, Oct 1999.

[30] Linda Northrop Paul Clements. *Software Product Lines - Practices and Patterns*. Addison-Wesley, 2002.

[31] K Pohl, G Böckle, and F van der Linden. *Software product Line Engineering - Foundations, Principles, and Techniques.* Springer, 2005.

[32] Dirk Muthig Patricia Costa Ralf Carbon, Mikael Lindvall. Integrating product line engineering and agile methods: Flexible design up-front vs. incremental design. In *1st International Workshop on Agile Product Line Engineering (APLE).*

[33] D.J. Reifer. How good are agile methods? *Software, IEEE*, 19(4):16–18, Jul/Aug 2002.

[34] I. Richardson and C. Gresse von Wangenheim. Guest editors' introduction: Why are small software organizations different? *Software, IEEE*, 24(1):18–22, Jan.-Feb. 2007.

[35] Colin Robson. *Real World Research- A Resource for Social Scientists and Practitioners-Researchers*, volume 2nd. Blackwell, 2002.

[36] Dale Churchett Ross Buhrdorf and Charles W. Krueger. Salions experience with a reactive software product line approach. In *Software Product-Family Engineering.*

[37] D. Sellier, M. Mannion, G. Benguria, and G. Urchegui. Introducing software product line engineering for metal processing lines in a small to medium enterprise. pages 54–62, Sept. 2007.

[38] Software Engineering Institute. *CMMI for Development, version 1.2*, 2006.

[39] Mikael Svahnberg and Jan Bosch. *Software Architectures for Product Families*, chapter Issues Concerning Variability in Software Product Lines, pages 146–157. Lecture Notes in Computer Science. Springer, Berlin / Heidelberg, 2000.

[40] Cooper K. Tian, k. Agile and software product line methods: are they so different? In *1st International Workshop on Agile Product Line Engineering (APLE).*

[41] Frank van der Linden. Family evaluation framework overview introduction. Technical report, ITEA project, 2005.

[42] Frank van der Linden, Klaus Schmid, and Eelco Rommes. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering.* Springer, 2007.

[43] Martin Verlage and Thomas Kiesgen. Five years of product line engineering in a small company. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 534–543, New York, NY, USA, 2005. ACM.

[44] Christiane Gresse von Wangenheim, Srgio Weber, Jean Carlo Rossa Hauck, and Gisele Trentin. Experiences on establishing software processes in small companies. *Information and Software Technology*, 48(9):890 – 900, 2006. Special Issue Section: Distributed Software Development.

Software Product Line Engineering Maturity Model for Small and Medium Sized Enterprises

Appendix I

Author: Siim Saarlo

Contents

1 Business Maturity Area

Business maturity area includes process areas that answer the question how to create profit with product line initiative. Process areas here deal with scope, costs, profits, market value and strategy among others.

B.1 Sales and Marketing

Sales and marketing should be involved in and influenced by the software product line engineering, so that these functions perform in synergy and mutually beneficial manner. Sales and marketing should not be based on single systems but product lines. The functions are aware of opportunities provided by SPLE and variability.
[10, 5, 4, 9, 30, 31, 3, 32]

- **B.1.1 Brand Name Strategy Aligned with SPLE**
  Organisation considers brand name as a driver of business success and has agreed on goals and activities that maximise the usage of SPLE opportunities in strengthening the brand. Multiple products from product line are collocated under one unified brand strategy.

  * B.1.1.ea1 Product Line Wide Brand Name **SPLEMM - 1**
    Different applications of product line are marketed under one brand name.

  * B.1.1.ea2 Defined Branding Strategy **SPLEMM - 1**
    Organisation has a documented and followed strategy to build, maintain and develop its brand. The strategy takes into account the limitations and benefits provided by SPLE.

- **B.1.2 Market Strategy Aligned with SPLE**
  Organisation has understanding about its market strategy and orientation. Market strategy stands for finding out what customer wants and offering it with competitive advantage. SPLE characteristics have distinct role in the selection of market strategy.

  * B.1.2.ea1 Using Process Qualities in Market Orientation **SPLEMM - 1**
    SPLE benefits of reduced cost and time to market are used to market products. User values of having a large amount of variability for low costs are defined as a main qualities of products. Organisation tries to gain customers by focusing on cost-leadership strategy.

  * B.1.2.ea2 Using Product Qualities in Marketing **SPLEMM - 2**
    SPLE benefits of increased usability and quality are used in marketing and promoted as main competitive difference.

  - B.1.2.ea3 Defined Product Definition Strategy **SPLEMM - 2**
    Product definition strategy determines who has the influence on defining product portfolio. Market situation, product line capabilities and future plans are considered when choosing product definition strategy. Generally customer driven mass customisation direction of strategy suits better with product line definition. Still in some cases other options could be considered to achieve specific goals. For example market or technology oriented producer-driven strategies.

  - B.1.2.ea4 Setting Pricing Policy with Respect to SPLE Specific Aspects **SPLEMM - 2**
    Prices and discount rates are calculated in cooperation between engineering and marketing. Fixed-price for systems can be set as customers are not willing to pay for work on SPL and core assets. For marketing purposes it may be considered to offer multiple products from product line with discount to customers.

  - B.1.2.ea5 Strategical Order of Entry Decisions **SPLEMM - 3**
    The right time is planned to launch a new application from product line in order to capture major shares of the market. Sometimes time-to-market requirements has to be relaxed in order to improve it for subsequent products. Products are strategically divided between different life-cycle stages to keep sustainable cash flows.

- **B.1.3 Relationship Management Aligned with SPLE**
  Communication and relationships with external stakeholders (eg. contacts with customers and suppliers) use and take into consideration the benefits and limitations of SPLE. Customer facing staff has proper overview of whole product line so it can represent its whole scale.

  - B.1.3.ea1 Centralised Customer Support **SPLEMM - 2**
    Instead of having separate customer support and servicing groups for different products on product line, SPLE characteristics like common user interfaces and higher quality are benefited from. Customer support is centralised and and provided by one team who has sufficient knowledge about all product line systems. Because of the commonalities between the products it is easier to achieve than among non-SPLE practitioners.

  - B.1.3.ea2 Managing Customer Interface **SPLEMM - 2**
    Processes and responsibilities are in place for customer interactions. It is important to consider the level and proficiency of people representing the organisation in front of customers. Processes and skills of people interacting with customers (support, sales, marketing) should support whole product line instead of single system. Also people interacting with customers have to

have skills and knowledge to recognise customer needs and opportunities for other product line applications.

* B.1.3.ea3 Collecting Feedback from Customers **SPLEMM - 2**
Feedback is collected from end users and used throughout whole product line. Data is collected and organised about the customers in a way that contributes into development of successive products.

* B.1.3.ea4 Limiting Customer Choices with External Variability **SPLEMM - 2**
Defined policy for limiting customer choices with external variability and for cases of exceptions in that policy. It must be made clear in the organisation and to external stakeholders what are the variation points in product line that customers can request change for. Variation points out of that list should be made invisible for customers.

* B.1.3.ea5 Active Involvement of Customers **SPLEMM - 2**
Customers are actively involved through conferences, user forums, beta testers etc. During that it should be tried to bring together users of different product line systems to increase mouth-to-mouth marketing and gather customers input into product development.

B.2 Scoping

Determine the products and the product features which should be included in the SPL. Main difference between SPLE and other reuse methodologies is that the product line is based on a clear vision of future products. This vision is initiated in scoping phase.
[7, 10, 11, 12, 17, 13, 18, 20, 3, 28]

- **B.2.1 Scope Planning**
Activities that include information collection and analyses necessary for defining product line scope. As SPLE is not considered suitable for totally new domains, then one of the first steps here should be analysing and understanding already existing product line.

  - **B.2.1.1 Collecting Information to Identify Domain Candidates for SPLE**
  Overview is established about the product line, its features, and how features are distributed in products. Different available sources has to be studied for this aim.

    * B.2.1.1.ea1 Studying Available Products in Existing Product Lines **SPLEMM - 1**

Characteristics of legacy and competitor products as well as future extensions to existing products are collected.

* B.2.1.1.ea2 Getting Information from Stakeholders **SPLEMM - 1**
  Small sized and new companies may lack the necessary domain knowledge. In these cases different stakeholders with information should be mapped and contacted. Stakeholders involve partners, suppliers, customers and also competitors.

* B.2.1.1.ea3 Using Information Sources Available in Organisation **SPLEMM - 2**
  Information produced by different functions in organisation is collected for scoping activities. This information includes for example relevant standards, knowledge gained through market analysis and technology forecasts.

– **B.2.1.2 Finding Domain Candidates**
  Analysing collected information to find potential domain candidates that could be suitable for SPLE. General SPLE standards and characteristics are considered in this initial selection instead of organisation specific goals and aims.

* B.2.1.2.ea2 Define Domain Candidates **SPLEMM - 1**
  Domain candidates are picked based on the simplest analyses of gathered information without using any specific tools.

* B.2.1.2.ea1 Specifying Product-Feature Matrix **SPLEMM - 1**
  Simple matrix to represent the information about product on product line and their common and variable features. This is a simplest tool to find potential candidates for SPLE domains.

– **B.2.1.3 Domain Candidates Potential Assessment**
  Domains with high reuse potential are identified through evaluation of sub-domains in the product line.

* B.2.1.3.ea1 Domain analysis **SPLEMM - 1**
  Assessment of domains and product line goals to identify domains with highest reuse potential. The assessment should be based on previously collected information. Form of the assessment is not specified.

* B.2.1.3.ea2 Identification and Prioritization of Product Line Goals **SPLEMM - 1**
  Product line goals should be defined and understood. Alignment between organisational goals and product line business goals should be established.

    * B.2.1.3.ea3 Technology Forecasting **SPLEMM - 2**
    Use technology forecasts for domain potential assessment. Understand how possible future development in technology influences product line.

    * B.2.1.3.ea4 Composing Initial Sketch of Business Case **SPLEMM - 2**
    Interview management and incorporate marketing and sales personnel into investigation of business potential of domain candidates. Perspective of achievement of product line goals should be understood with this step.

    * B.2.1.3.ea5 Using Domain Experts **SPLEMM - 2**
    Individuals with knowledge and experience with candidate domains should be identified and incorporated into analysis. The knowledge of in house or external domain experts should be used to achieve thorough understanding of domains.

    * B.2.1.3.ea6 Using Formal Methods for Domain Analyses **SPLEMM - 3**
    Wide selection of formal methods exist for domain analysis : Scope, commonality, and variability (SCV) analysis, Domain analysis and design process (DADP), Feature-oriented domain analysis (FODA), Synthesis process of the reuse-driven software processes (RSP) approach, Domain analysis process of organizational domain modeling (ODM), Product Line Software Engineering Customizable Domain Analysis (PuLSE-CDA), Domain-Specific Modeling (DSM).

- **B.2.2 Scope Maintenance**
  Output of scope planning activities is stored and used in further SPLE activities.

  – B.2.2.ea1 Definition of Scope **SPLEMM - 1**
  Document decisions made in scope planning sub-process. Describe domains and scope and avoid both over generalisation and over trivalisation when doing that. Also stakeholder concept overlap is common mistake that and should be avoided from happening.

  – **B.2.2.2 Definition of Systematic Scoping Processes**
  Product line scoping should be systematic and ongoing process in order to understand domain and foresee future requirements. Strategic discussions at the level of features and requirements centered on the product line scope should be performed to continuously test the product line scope.

    * B.2.2.2.1.ea1 Developing Product Line Scenarios **SPLEMM - 2**
    Extensive analysis of future domains and requirements in regular

bases should result in scenario document that is used and kept up to date.

* B.2.2.2.1.ea2 Defined Scope Management Processes **SPLEMM - 2**
  Explicit and periodic scoping and road-mapping processes are put in place.

* B.2.2.2.1.ea3 Adopting Comprehensive Scoping Process **SPLEMM - 3**
  Organisations with high maturity and processes with growing complexity may consider improving and standardising their scoping procedures based existing methodologies in industry. For example PuLSE-Eco is one of the few available comprehensive scoping methodologies.

B.3 Business Planning

SPLE should be reflected in organisations vision, business objectives and also traced by financial management.
[5, 4, 25, 21, 30, 32, 3]

- **B.3.1 Strategic Planning**
  Organisation wide planning of the software product family business and development on strategic level.

  - **B.3.1.1 Definition of Strategic Goals and Plans**
    Organisation has long term strategy and plans that are in alignment with SPLE. Characteristics added to organisation by SPLE are considered and used on strategic level. SPLE specific characteristics are included and considered in strategy building process.

    * B.3.1.1.ea1 Analiese of Internal and External Environment, Risks and Opportunities **SPLEMM - 2**
      Usage of different tools helps organisation to get a thorough picture of different aspects that affect their product line performance. SWOT and PEST analyse are examples of such tools. Relations among systems of product line are considered in environment analyse. Overview of possible risks and opportunities is established during the analyse.

    * B.3.1.1.ea2 Using Product Roadmaps **SPLEMM - 2**
      Goals and results of domain engineering are considered on strategic level using roadmaps of future products. Product roadmaps give a perspective of different products and give an opportunity to build connections between the product line level and strategic level.

    * B.3.1.1.ea3 Distinguishing Between Domain and Application Planning **SPLEMM - 2**
      There are separate plans and roadmaps for domain and application engineering. The plans are related and commonalities in applications provide the basis of the domain engineering plan.

    * B.3.1.1.ea4 Defined Portfolio Management Processes **SPLEMM - 3**
      There is defined process for utilising plans and roadmaps of product line and its applications, which are coordinated and used strategically to gain best business value out of SPLE.

  – **B.3.1.2 Business Vision**
    SPL should have a big role in business vision and the vision should address the benefits and reflect the nature of SPLE. The vision should motivate to reach strategic goals through proper SPLE practices.

    * B.3.1.2.ea1 Using Vision as Decision Making Tool **SPLEMM - 1**
      Business vision is in practical use as decisions on different levels and functions are measured against it
    * B.3.1.2.ea2 Incorporating SPLE in business vision **SPLEMM - 2**
      SPLE has the central part in business strategy. It is recognised that strategic goals and development towards vision are achieved through practicing SPLE. Foreseeing SPLE vision responds to new market needs and changing technology
    * B.3.1.2.ea2 Quantitatively Incorporating SPLE in Business Vision **SPLEMM - 3**
      SPLE, its value and evolution is incorporated in quantitative way in business objectives and vision. Advantages of SPLE appear in vision and business objectives and at the same time the drawbacks are discovered and their effects diminished on strategic level. Vision and business objectives are related to SPL development upon a well-understood bases.

- **B.3.2 Budgeting and Investment**
  Integration of SPLE related aspects in budgeting, financial management and funding activities.

  – **B.3.2.1 Financial Management of SPLE**
    SPLE is reflected in financial management of the organisation.

    * B.3.2.1.ea1 Establishing Budget for Domain Engineering **SPLEMM - 2**
      Investments are made and budgeted for domain-engineering activities and for repository of reusable assets. Costs for example for refactoring of reference architecture are included in budget.

* B.3.2.1.ea2 Managing Domain Engineering Budget **SPLEMM - 3**
  Mechanisms in place to generate budget for domain engineering by results of sales. Costs and savings of reuse and variability and SPLE is measured, and reflected in budgets. In more advanced organisations budgeting and investment are accurately integrated with the forecast of sales, costs and savings of SPL products.

– **B.3.2.2 Financial Management Involvement in Product Line Development Decisions**
  Financial justification is provided for the choice of products and the product line approach to build them.

  * B.3.2.2.ea1 Prediction of Future Costs and Benefits Using Current Development Approach **SPLEMM - 2**
    Financial management has an ability to predict costs and benefits organisation has when continuing with current development approach

  * B.3.2.2.ea2 Prediction of Future Cost and Benefits of Product Line Approach **SPLEMM - 3**
    Financial management has an ability to foresee costs and benefits organisation has when using product line approach.

  * B.3.2.2.ea3 Investment Analysis about Product Line Related Costs **SPLEMM - 3**
    Financial management has a capacity to predict costs related to improving product line or converting to product line approach.

2 Domain Engineering Maturity Area

Domain Engineering is a maturity area that deals with development of core assets base (reference architecture). The area follows development of core assets through the traditional software development life-cycle stages.

D.1 - Domain Requirements Engineering

Development, management and documentation of common and variable domain requirements.
[3, 1, 27, 8, 9, 10, 19, 22, 23, 26]

- **D.1.1 Requirements Elicitation**
  Process of collecting stakeholder needs and translating them into requirements. The goal is to cover maximum amount of requirements of all envisioned product line applications.

  - **D.1.1.1 Using Different Sources of Requirements**
    Involvement of potentially large number of different sources for requirement identification.

    * D.1.1.1.ea1 Involving Different Roles **SPLEMM - 2**
      Roles like product managers, architects, customer groups, maintenance staff etc. should be involved in requirement elicitation process.

    * D.1.1.1.ea2 Involving Different Sources **SPLEMM - 2**
      Other potential sources involve legacy systems or external sources like country laws, environment and safety regulations etc.

    * D.1.1.1.ea3 Studying Existing Applications **SPLEMM - 2**
      Already existing applications can be reviewed as a source of requirements for product line.

  - **D.1.1.2 Initial Organisation of Requirements**
    Organise collected requirements for further analyse and usage. Because of the high number and growth of requirements they should be organised in a manner suitable for concrete case.

    * D.1.1.2.ea1 Prioritisation of Requirements **SPLEMM - 1**
      Priorities should be organised into arbitrary, named subsets or concrete release strategy by prioritising the software requirements and mapping them to future releases of the software.

    * D.1.1.2.ea2 Separation of Problem and Solution Space **SPLEMM - 1**
      Separate pure customer requirements from technical requirements.

    * D.1.1.2.ea3 Defining Requirements Organisation Process **SPLEMM - 2**
      SPLE needs more rigid requirements organisation process because number of requirements tend to be high.

    * D.1.1.2.ea4 Categorise Requirements **SPLEMM - 2**
      Depending on development procedures it might be beneficial to organise requirements based on their types. Keeping the bug-reports, enchancement requests, smaller new features requests etc in separate group may become beneficial when organisation has special debugging iterations. This also makes it easyer to decide if something should be approached on reference architecture level or on single applications.

    * D.1.1.2.ea5 Using Proven Representation Methods **SPLEMM - 3**
      Example of such methods is definition hierarchies that organises the requirements into a definition hierarchy and shows requirements for different products in the same hierarchy. Chosen or developed method should identify architectural drivers of the product family and show how different products in the family vary. Other example of such method is to organise requirements into Lucent's Commonality Analysis document

- **D.1.2 Commonality and Variability Analysis**
  Elicited requirements should be analysed to understand the required variability in them. This makes it possible to design required variation points during architectural and detailed design.

  – **D.1.2.1 Identify a Set of Common and Variable Requirements**
    In application requirements engineering common requirements must be transferred to domain level. In domain engineering process area elicited requirements have to be analysed to find the ones that are specific to only some applications and should be implemented as variants in domain variability points. Less common requirements can be left to be specified during application engineering.

    * D.1.2.1.ea1 Application-Requirements Matrix **SPLEMM - 1**
      Simple way to keep track on requirements and their existence in applications of product line by visualising them in the 2-dimensional table.

    * D.1.2.1.ea2 Priority-Based Analysis Scheme **SPLEMM - 2**
      Kano model is an example of priority based analysis scheme. This helps to organise customer requirements into different groups depending on their effect on customer satisfaction with product.

    * D.1.2.1.ea3 Check-List Based Analysis **SPLEMM - 2**
      Keeping track of categories of potentially common requirements like strategic commonalities, country laws, customer's basic needs etc.

  – **D.1.2.2 Requirements Variability Documentation**
    Variability must be documented already in requirements. Effective documentation guarantees that people in marketing and development understand each other and can agree on implementation.

    * D.1.2.2.ea1 Defining Variation Points and Variants in Requirements **SPLEMM - 1**
      Define and document variation points and variants with their dependencies in requirements

    * D.1.2.2.ea2 Distinguish Between Internal and External Variability **SPLEMM - 1**
      Distinguish between variability that is only visible to engineers and variability for customers

    * D.1.2.2.ea3 Document the Common Requirements in Detail **SPLEMM - 2**
      Requirements documentation process and methods are agreed in the organisation. These allow representation of requirements and back-tracing links between requirements and following design and implementation issues. Feature models, use-case models and functional models have been suggested for requirements documentation in SPLE cases

- **D.1.3 Requirements Management**
  Elicited requirements have to be collected, organised and used in a way that keeps their quality high.

  – **D.1.3.1 Manage Requirement Changes**
    Requirements may change in time. Collected requirements have to be managed in a way that allows to reflect those changes in documented requirements. In case of SPLE extra attention has to be payed to keep variants up to date.

    * D.1.3.1.ea1 Communicate Software Requirements **SPLEMM - 1**
      Establish communication mechanisms for dissemination of software requirements, and updates to requirements to all parties who will be using them. In SPLE environment requirements have to be discussed intensively across larger group of people. By default it is one of the responsibilities of architect. This can be done by relatively informal discussion support integrated in

processes.

  * D.1.3.1.ea2 Process for Integrating New Requirements **SPLEMM - 2**
    Defined process have to exist to integrate future requirements into product line. This process should include regular analysis of possible future product line requirements that take into consideration business strategy, software industry development and customer needs.

  * D.1.3.1.ea3 Receiving Feedback from Application Engineering Cycles **SPLEMM - 2**
    New requirements or changes in requirements have to get from application engineering to domain engineering through feedback cycles. Organisational issues should be considered when establishing processes.

  * D.1.3.1.ea4 Reactive Customisation of Product Line Requirements **SPLEMM - 1**
    With smaller companies and new markets insufficient domain knowledge can be a problem. Reactive customisation of product line requirements may be used in this case until enough domain knowledge is gathered.

- **D.1.3.2 Guarantee Quality of Requirements**
  Evaluate the consistency and establish traceability between software requirements and system requirements. Extra effort has to be put on quality of common requirements as they affect all the systems derived from product line. In later stage of SPL life it can happen that more focus is put on architectural issues which may loosen the consistency between product line and its requirements.

  * D.1.3.1.ea1 People Carrying Knowledge **SPLEMM - 1**
    Easiest way to make sure that customer requirements are understood correctly in design phases is to involve some people in both of the processes. Architects should be part of requirements elicitation processes to bring the understanding and knowledge of requirements logic into design. Also it may be beneficial to integrate architects into customer negotiations.

  * D.1.3.2.ea2 Definition of Requirements Management Process **SPLEMM - 2** Requirements engineering process should be documented properly so that all the stakeholders have same understanding of how it works and what are the standards to be followed when creating and managing artefacts in the process.

  * D.1.3.2.ea3 Document Assumptions of Commonalities **SPLEMM - 2** Reasons and assumptions on why concrete requirement is

common should be documented. This allows to validate those assumptions in further stages of product line.

* D.1.3.2.ea4 Compromise Between Quality and Generality **SPLEMM - 2**
  In SPLE contexts quality requirements has to be lowered sometimes to gain in generality. Organisation has to be aware of the trade-offs and have common policy for decisions on these issues.

* D.1.3.2.ea5 Regular Roundtable Meetings **SPLEMM - 2**
  Regular meetings between requirements engineers, lead architects, and marketing and sales people are held in order to balance requirements and architecture.

* D.1.3.2.ea6 Usage of Defined Requirement Management Tools or Processes **SPLEMM - 3**
  Complexity of requirements management in SPLE often requires creation of customised tools or processes to support standard methodologies. Also some dedicated processes for SPLE requirements engineering are available. Pulse-CDA is an example of pre-defined process.

D.2 - Domain Architecture Definition

Process area where product line core assets are designed and realised.
[3, 1, 7, 9, 10, 17, 25, 15, 23, 29, 32]

- **D.2.1 Develop Architectural Design**
  Initial decisions about domain architecture design
  - D.2.1.ea1 Defining Production Strategy **SPLEMM - 2**
    Defining, choosing and using the variation mechanisms provided by the architecture. That strategy describes how the organisation plans to build the specific products from the core assets.
  - **D.2.1.1 Deciding on Usage of Architectural Styles and Patterns**
    Some high level decisions has to be made on the initial design of reference architecture. These also have to be reviewed continuously and communicated. Some styles and methods are proven and more common in designs of product line architectures, and therefore should be considered as first-hand options.
    * D.2.1.2.ea1 Aspect-oriented Software Development (AOSD) **SPLEMM - 2**
      Allows to eliminate causes of scattering and tangling of features by a good separation of concerns (features)

    * D.2.1.2.ea2 Usage of Layered Architecture Style and Strategy Pattern **SPLEMM - 2**
      Engineers should separate context-specific, domain-specific, and product category specifics. Layered architecture style and strategy pattern are example of available opportunities to separate different types of functions

- **D.2.2 Define Component Interfaces**
  Development and implementation of a design for the external and internal interfaces.

  – D.2.2.ea1 Interface Design **SPLEMM - 1**
    Interface design process has to incorporate all developers of related components (providing or requiring). The design is a compromise between the abilities of components that provide or use the interface functionality.

  – D.2.2.ea2 Interface Implementation **SPLEMM - 1**
    Interface elements have to be declared in programming language file and added to implementations of componentes that provide or require the interfaces.

- **D.2.3 Define Connecting Components**
  Development and implementation of a design for core asset components.

  – **D.2.3.1 Design for Variability and Quality Requirements**
    Design, implementation and general usage of the variation points and general mechanisms for achieving variability in a product line architecture and in consisting components. Chosen variation mechanisms must support variations required by products, pre-defined production strategy and efficient integration of new products. Besides variability product line development demands higher quality in terms of variability, flexibility, evolvability and maintainability. There are some proven design methods that help to achieve these criteria

    * D.2.3.1.ea1 Separation of the Common and Variant Behaviour **SPLEMM - 1**
      Good design requires distinct separation between common parts and variation points in product line architecture components.

  – **D.2.3.3 Make/Buy/Mine/Commission Analysis**
    New development should occur only after carrying out the activities in the *Make/Buy/Mine/Commission Analysis* practice area. Decision has to be made between building a new component in house, buying available 3rd party products or letting 3rd party to build a necessary component. Also existing systems with good architecture, design and implementation can be used for creating a baseline for

the core assets - mining. For example COM interface can be used to connect legacy system which is written in other language than new one.

* D.2.3.3.ea1 Defined Standard Activity **SPLEMM - 1**
  No analyse is performed for separate components or component groups. Organisation uses default strategy for all components. Irregular add-hoc decisions may happen sometimes.

* D.2.3.3.ea2 Breath-First Analysis of Options **SPLEMM - 2**
  The most basic way of weighting all the options by using predefined sets of questions. Questions should compare the costs and benefits, experience with the option and other relevant decision factors. Questions should be organisation specific.

* D.2.3.3.ea3 Using More Sophisticated Techniques and Tools **SPLEMM - 3**
  The growth of the product line can be supported for example with SEI's OAR/SMART practice for minging existing assets and CoVAR process for selecting 3rd party components (COTS). Also some commercially available tools exist on the market.

- **D.2.4 Domain Realisation**

  - **D.2.4.1 Component implementation**
    Implementation of component designs in programming language.

    * D.2.4.1.ea1 Usage of Agile Principles **SPLEMM - 2**
      Despite some differences in approaches agile principles and SPLE may be suitable combination. Specially in SME environments it is beneficial to leave the development process agile after SPLE adoption.

    * D.2.4.1.ea2 Usage of Test-Driven Development **SPLEMM - 2**
      Unit testing is a common and important practice in SPLE that holds the roles of initial testing and also protects the quality of reference architecture from changes in future.

    * D.2.4.1.ea3 Encapsulation of Legacy Systems **SPLEMM - 2**
      Wrapping legacy systems and integrating them to SPL has proven as highly beneficial practice to lower the costs of SPLE practice initiation.

  - **D.2.4.2 Compilation**
    There exists a process for compiling components into object files that can be linked into working executables during application realisation.

    * D.2.4.5.ea1 Automating Compilation Activities **SPLEMM - 2**
      Compilation activities should be integrated with testing and ref-

erence architecture organisation. Automated process should involve compilation, binding and unit testing. There exist examples of Fowler's Cruise Control usage in SPLE settings.

- **D.2.5 Domain Architecture Evaluation**
  Evaluation of reference architecture can be organised in different forms. What ever is the choice of methods it is important that quality of the architecture is guaranteed through that step.

  - **D.2.5.1 Review-Processes for Product Line Quality Requirements**
    Extra attention should be paied to product line specific quality requirements like variability, flexibility, evolvability, and maintainability to assure quality and sustainability of the product line. Several architecture review techniques exist that can be used in product line context. These techniques are categorised as questioning and measuring techniques. Often mechanisms like Java reflection and late binding is used in product lines. In these cases static code analysis are not sufficient but dynamic analysis using runtime information should be used instead.

    * D.2.5.1.ea1 Architecture design reviews **SPLEMM - 2**
      Rather informal design review meetings. Exact form and participants can be decided by the organisation. For example these can be run in a form of peer and team reviews.

    * D.2.5.1.ea2 formal architecture evaluation methods **SPLEMM - 3**
      Usage of more formal methods could be considered for mature architectures. Example of such methods include SEI Architecture Tradeoff Analysis Method (ATAM) or the SEI Software Architecture Analysis Method (SAAM), software performance egnineering (SPE)

    * D.2.5.1.ea3 Establishing Arhitecture Review Board **SPLEMM - 3**
      Architecture review board should be responsible for the overall architecture, its quality and general accordance to design principles. Also the decisions about how and where to implement requirements can be overviewed by the board. It also has the communicational responsibility to mediate across product and platform development.

  - **D.2.5.2 Refactoring**
    Refactoring processes is necessary to keep asset evolution under control.

  - D.2.5.2.ea1 Ad-hoc Refactoring **SPLEMM - 1**
    Non planned refactoring happens irregularly

  - D.2.5.2.ea2 Including Refactoring Activities in Processes **SPLEMM - 2**
    Refactoring is incorporated in standard architecture development processes. Refactoring costs are accepted and importance understood even when customer is not paying for it directly (higher level of maturity).

  - D.2.5.2.ea3 Removing Unused Component Parts **SPLEMM - 2**
    Removing of the component parts that are not used to avoid dead code.

  - D.2.5.2.ea4 Finding Emerging Abstractions **SPLEMM - 2**
    Usage of customisation and refactoring to find emerging abstractions in order to avoid inaccurate and over-generalized proactive design efforts.

  - D.2.5.2.ea5 Incorporating Business Unit Architectures **SPLEMM - 2**
    Business unit architectures are incorporated in code reviews and refactoring to minimise dependencies between reusable assets.

  - D.2.5.2.ea6 Minimising Interfaces **SPLEMM - 2**
    Refactoring and redesign of product line and its assets should aim at minimising the interfaces between the components. This mitigates the problem of architecture with growing complexity.

D.3 - Domain Testing

Domain testing looks for defects in domain artefacts and creates reusable test artefacts for application testing. Domain testing processes are generally same as single-systems but additionally has to deal with variability and the issue of non-existing executable system.
[3, 23, 32]

- **D.3.1 Choice of Test Strategy and Approach**
  Assets to be tested are analysed and suitable test cases are identified for separate assets. In theory there are two extreme strategies for quality assurance: product-focused and infrastructure-focused. One is performing the tests on derived applications and the other on core assets respectively. Because of the flaws in both of these, some less compromising strategies are used in practice.

  - D.3.1.1.ea1 Using a Sample Application Strategy (SAS) **SPLEMM - 1**
    Sample applications are used to test domain artefacts or finalised

platform development stalled until first application is ready to be used for testing instead of sample application.

    * D.3.1.1.ea2 Using Commonality and Reuse Strategy (CRS) **SPLEMM - 1**
    Common parts of core assets are tested during domain testing. At the same time reusable test artefacts are prepared for variable parts and these are used in application testing phase when variabilities are binded.

    * D.3.1.1.ea3 Composite Strategy **SPLEMM - 1**
    This strategy combines strengths of SAS and CRS strategies by enforcing the creation of reusable test artefacts in domain testing and the reuse of these artefacts in application testing.

- **D.3.2 Domain Test Specification and Construction**
  Building process of test artefacts.

  – D.3.2.1.ea1 Creating Logical Test Cases **SPLEMM - 1**
  Test cases without concrete details like data, GUI elements etc are created.

  – D.3.2.1.ea2 Creating Detailed Test Cases **SPLEMM - 1**
  Logical test cases are improved with details. Detailed test cases are only created for common parts of reference architecture.

  – D.3.2.1.ea3 Creating Traceability Links **SPLEMM - 2**
  Creating traceability links between test artefacts and test references. Traceability links allow to reuse test artefacts in application engineering.

- **D.3.3 Domain Test Execution and Evaluation**
  The phase of conducting designed tests and analysing results

  – D.3.3.ea1 Domain Unit Testing **SPLEMM - 2**
  Use rigid unit testing (eg. Junit framework) to avoid problems with core assets usage. If variability is realised then the unit has to be built with each defined variant. From then each build is tested as in single-system engineering.

  – D.3.3.ea2 Integration Testing on Domain Engineering Level **SPLEMM - 2**
  In SPLE context variability makes it impossible to test all component interactions during domain testing. It is more reasonable to test only common interactions and those that contain few variable interactions with already realised components.

  - D.3.3.ea3 System Testing on Domain Engineering Level **SPLEMM - 2**
    System tests are influenced by variability that is defined in domain requirements artefacts. As parts of the system tested in this level are typically large, it is hard to find test cases without variability. Thus system tests can be performed with a defined configuration of variants.

  - D.3.3.4 Documenting Test Cases **SPLEMM - 2**
    Test cases has to be documented in order to guarantee verifiability and repeatability of tests. For that test protocol including test case, version number of the tested object and the test result should be created. Test records and error classes should be analysed and origins of the errors determined. Each test run should have test summary raport.

D.4 - Product Line Architecture Documentation

Documentation should aim to give thorough overview of architecture and available assets and visualise dependencies between assets. Architecture should be described in well known notations such as UML, covering all relevant architectural views and using clearly defined semantics.
[3, 10, 9, 23]

- **D.4.1 Documentation Processes Management**
  Management practices related to documentation.

  - D.4.1.ea1 Assigning Role for Architecture Documentation Responsible **SPLEMM - 1**
    Responsible for documentation must be set to guarantee high quality. Also documentation should have an essential part of development processes and core assets.

  - D.4.1.ea2 Documentation and Guidelines for Application Builders **SPLEMM - 1**
    Descriptive scenarios have to be provided to application builders that explain the features of reference architecture.

  - D.4.1.ea3 Communicate Architecture **SPLEMM - 1**
    Architecture should be communicated to different stakeholders through appropriate documentation. It has to be made sure that business and marketing people, who might not understand technical notation, have a possibility to understand it. Also it has to be quaranteed that different stakeholders get the information they actually need.

- **D.4.2 Architecture Documentation**
  Proper documentation techniques for product line architecture.

- D.4.2.ea1 Using Architectural Scenarios **SPLEMM - 1**
  Describe architectural scenarios that present the architecture from a system use perspective. Use cases and textual descriptions can be used for that.

- D.4.2.ea2 Using Different Views to Represent Architecture **SPLEMM - 2**
  Different views can be used to guarantee good visibility of architecture. Possible views include a *module-decomposition view*, a *communicating-processes view*, a *layered view*, a *deployment view*.

- D.4.2.ea3 Specifying Component Interfaces **SPLEMM - 2**
  Use contractual approach, state machines, interval temporal logic and aim for minimising the interfaces between components to simplify the architecture

3 Application Engineering Maturity Area

Application Engineering maturity area consists of software engineering activities that are related to production of product line applications. It ivolves application specific requirements elicitation and development, application deriviation from core assets, and application testing.

A.1 Application Requirements Engineering

The goal of application requirements engineering in SPLE is to elicit and to document the requirements specific for a particular application and at the same time use the domain requirements artefacts as much as possible.
[31, 23, 3]

- **A.1.1 Requirements Elicitation** The process of application specific requirements elicitation and reference requirements reuse.

  - **A.1.1.1 Elicitation of Application Specific Requirements** Requirements are collected from application stakeholders. It is important to get information from all relevant stakeholders that were not covered during domain requirements engineerig. Also the ones already interviewed there should be considered if exists a chance that they have specific needs towards the application in question.

    * A.1.1.1.ea1 Analysing Target Group Needs **SPLEMM - 1**
      Elicitation of application specific requirements should start with identification of the application target group and analyzation of their needs. Methods from common single-system development practices can be used.
    * A.1.1.1.ea2 Using Subject Matter Experts **SPLEMM - 2**
      Specific expert knowledge should be used also during application specific requirements elicitation besides scoping and domain requirements elicitation phases. Relationships with proper expertise sources should be mapped and managed.

  - **A.1.1.2 Domain Requirements Reuse**
    Application engineering should maximise the reuse of reference requirements artefacts.

    * A.1.1.2.ea1 Communication of External Variability to Stakeholders **SPLEMM - 1**
      Stakeholders should have clear understanding of available variability choices supported by reference architecture. External variability is this part of supported variability in reference architecture that is available for customers to change. The more customers make their change requests among these variables, the higher is reuse of reference requirements artefacts.

    * A.1.1.2.ea2 Usage of Domain Variability Model **SPLEMM - 1**
      Domain variability model is documentation of supported variability in reference architecture. It can be used to better communicate product line capabilities to stakeholders.

- **A.1.2 Application Requirements Analyse** Requirements that are collected during Application requirements elicitation have to be analysed to distinguish the ones that could be common throughout whole product line and should be included to reference architecture. Also in this stage it should be decided weather the product is viable to be developed as a part of a product line or not.

  – **A.1.2.1 Distinguish Between Product Line Wide and Application Specific Requirements**
    Product line wide requirements can arise during application requirements analyse. These requirements has to be effectively communicated back to domain engineering.

    * A.1.2.1.ea1 Review Application Specific Requirements **SPLEMM - 1**
      Collected application specific requirements have to be reviewed to find potential domain wide requirements
    * A.1.2.1.ea2 Defined Decision Criteria Exists for Domain Wide Requirements **SPLEMM - 2**
      There exists concrete process for deciding if any requirement should be considered as domain wide and suggested to add as a core asset.

  – **A.1.2.2 The Evaluation of Realisation Effort for Requirement Deltas**
    Reuse of core assets is maximised by using domain requirements to satisfy stakeholder requirements where possible. Stakeholder requirements that do not correspond to domain requirement artefacts are application specific or can apply for more than one application. In latter case it might be beneficial to implement as requirement delta. These requirements has to be analysed to decide weather the delta should be realised in the application or not.

    * A.1.2.2.ea2 Analysing Requirement Deltas **SPLEMM - 2**
      Analyse requirement deltas with respect to variability model, domain requirements artefacts, and the application architecture.

    * A.1.2.2.ea1 Deciding about Requirement Deltas **SPLEMM - 2**
      Trade-off decisions made about which of the stakeholder requirements to include as application requirement.

  – **A.1.2.3 Decision about Application's Viability as a Product Line Member**

Application specific requirements help to estimate the cost of the product and its feasibility for production. These estimations give a base to application's business case.

  * A.1.2.3.ea1 Comparing Application Requirements to Product Line Scope Definition **SPLEMM - 1**
    The control should be performed if the application in question fits into the criteria of product line scope.

  * A.1.2.3.ea2 Updating Product Line Scope Definition **SPLEMM - 2**
    It may happen that customer demands for many products that are slightly out of scope of product line. This may indicate that scope definition should be extended. Trade-offs between changing the scope and rejecting customers requests must be considered and proper actions taken in these cases.

- **A.1.3 Documentation of Application Requirements**
  SPLE specific aspects and issues are considered when documenting application requirements

  – A.1.3.ea1 Establishment of Traceability Links **SPLEMM-2**
    Record traceability links between the domain requirements artefacts and application requirements artefacts.

  – A.1.3.ea2 Documentation of Variability Bindings **SPLEMM - 2**
    Application variability documentation must document the bindings of the variation points defined in the domain variability model.This means there is a clear way of indicating which variant is chosen in each variability point for a application in question.

A.2 Application Design

Design of the application specific architecture and usage of design artefacts from reference architecture.
[25, 16, 31, 23, 24]

- **A.2.1 Application Specific Modelling** Creating the design of applicatoin by adding application specific abstractions and models to reused reference architecture artefacts.

  – A.2.1.ea1 Establishing Application Specific Design Documentation **SPLEMM - 1**
    Extracting application design documentation from core assets and updating it with application specific abstractions and models.

  - A.2.1.ea2 Product Line Principles are Used in Application Design **SPLEMM - 2**
    It is good practice to also use general domain design principles in application design besides tangible core assets. Attention should be payed to this point specially when there are separate teams for core assets and application development. Usage of established product line texture helps to meet this goal.

- **A.2.2 Mining / Accessing Appropriate Core Asset**
  Duplication of core assets in application engineering should be avoided by maximising the usage of core assets.

  - **A.2.2.1 Mining Existing Assets for Application Development**
    In order to take maximum out of reusable assets provided by reference architecture, application engineering has to have good overview about asset base and defined processes for finding and selecting necessary assets.
    * A.2.2.1.ea1 Utilisation of the Organisation of Core Asset Base **SPLEMM - 1**
      Reused domain requirements are used as a first hint of opportunities for reuse as common requirements are implemented in core assets. Traceability links allow to find implementation assets of domain requirements. Generally speaking, application engineers have to have overivew about organisation of core assets.

  - **A.2.2.2 Using Existing Assets in Application Development**
    Domain assets are reused in application development by incorporating them into application design.
    * A.2.2.2.ea1 Limiting Further Development of Core-Asset Based Products **SPLEMM - 2**
      Products generated from core assets should not be developed further to keep the architectural stability among whole product line and benefit from effective configuration management.

- **A.2.3 Binding of Variants**
  Variation points in core assets are bound to application specific variants according to bindings.

  - A.2.3.ea1 Selecting Suitable Variants in Variation Points **SPLEMM - 1**
    Reasoned decisions guarantee consistent selection of component variants. (SPLE book)
  - A.2.3.ea2 Binding of Variants **SPLEMM - 1**
    Variants of reference architecture's variation points are bound based on application variability model. Traceability between variability in domain requirements artefacts and the reference architecture is used

to accomplish the process.

- **A.2.4 Feedback from Application to Domain Engineering Life-Cycle**

  - A.2.4.ea1 Integrating Application Specific Elements to Reference Architecture **SPLEMM - 2**
    Application specific artefacts that are considered reusable for other applications are integrated into domain artefacts by domain architect (or other role responsible for the task).
  - A.2.4.ea2 Updating Application Design with Reusable Components **SPLEMM - 2**
    When application artefact has been integrated to core assets, the application should use new version of reusable artefact instead of original application specific one. This allows to reduce the amount of application specific artefacts that have to be maintained.

A.3 Application Realisation

Provides detailed design and implementation of application-specific components and interfaces. Selected variants of reused components, and application configuration. Final outcome is a working application that is ready for testing.
[25, 31, 23]

- A.3.ea1 Configuration **SPLEMM - 1**
  Applications that are derived from core asset base should reuse these assets as much as possible. Product derivation should be supported by automated configuration tools or techniques which allows to minimise the work on application engineering.

- A.3.ea2 Realisation of Application-Specific Components **SPLEMM - 1**
  Similar activities to realisation of single systems but lot of effort is put on reuse of product line opportunities. Domain interfaces should be used where possible to keep the common structure even on application specific parts. Also the design of existing variants should be used. For example in case of application component and reusable component being variants of same variation points it is suggested to use the design of existing variants as an input for the design of the new variant.
- A.3.ea3 Building the Application **SPLEMM - 1**
  As a final task of realisation the application is built using organisation standard process. This activity includes compiling, linking and deploying of the software. Depending on used configuration mechanism variation points of reused components are bound in some of these steps.

A.4 Application Testing

Application testing process area is a step necessary to achieve a sufficient quality of the application under test. The tests cover both reusable and application specific components. Thus the area complements the testing activities of domain testing. Besides some more common quality assurance activities like code inspections, static analysis of source code and manual testing some SPLE specific issues has to be considered.
[3, 14, 23]

- A.4.ea1 Handling SPLE Specific Aspects in Testing **SPLEMM - 1**
  Application tests have to handle variability issues in tests. For that variant absence and application dependency tests could be useful.

- A.4.ea2 Domain Test Artefact Reuse **SPLEMM - 2**
  Core assets include test artefacts that should be used during application testing. Depending on testing methodology (Component Reuse Strategy (CRS) or Sample Application Strategy (SAS)) artefacts available for reuse may differ.

4 Collaboration Maturity Area

Collaboration is a maturity area that assembles miscellaneous processes that are interacting with both domain and application engineering areas. It includes different techical management areas, organisational structure and communication issues.

C.1 - Variability management

The explicit use of variation points and supporting mechanisms. Involves variability management processes that support and control variability utilisation in domain and application engineering paradigmas. Variation points and variants allow to delay design decisions. Thereby it is possible on domain level to mananage differences between applications.
[3, 6, 12, 17, 19, 30, 23, 29]

- **C.1.1 Identification of Variation Points**
  Identification of variation points is an ongoing process in SPLE. Besides variability identification in domain and application engineering maturity areas, it should be incorporated as continuous part of general processes.

  - C.1.1.ea1 Incorporating Ongoing Variability Points Identification in Processes **SPLEMM - 2**
    Organisation wide processes support an ongoing identification of variation points. It should get input from new application engineering, ongoing scoping and other sources of relevant information.

  - C.1.1.ea2 Evolution of Variability **SPLEMM - 2**
    Evolution of variability may happen through introduction of a new variation point, changed binding time, changed variant addition time, changed variation point dependences, removed variation point or added variants. Organisation has defined process in parallel with variability identification to control and manage the evolution of variability.

- **C.1.2 Variability support processes**
  Support for variability decisions in engineering areas should be incorporated in organisation wide processes.

  - **C.1.2.1 Evaluating Trade Offs Between Early and Late Binding**
    When late binding of variants increases the flexibility of product line, it still has several trade offs including higher resource cost, decrease in testability and thus predictability of quality, increased need for development resources and problems with implicit context dependencies. Organisations should possess knowledge and preferences about the issue.

* C.1.2.1.ea1 Choice of Binding Time for Variability Implementation **SPLEMM - 1**
  Choice is made considering all different possibilities and their pros and cons. Different technologies allow to set the binding time before compilation, at compile time, at link time, at load time or at Run-time.

* C.1.2.1.ea2 Choice of Variability Mechanisms **SPLEMM - 1**
  Depends on the choice of binding time. Examples of variability mechanisms include [29]:
  a) Inheritance, is used when the variation point is a method that needs to be implemented for every application, or when an application needs to extend a type with additional functionality.
  b) Extensions and extension points, is used when parts of a component can be extended with additional behaviour, selected from a set of variations for a particular variation point.
  c) Parameterisation, templates and macros, are used when unbound parameters or macro expressions can be inserted in the code and later instantiated with the actual parameter or by expanding the macro. For example templates and ifdefs in C++.
  d) Configuration and Module Interconnection Languages, are used to select appropriate files and fill in some of the unbound parameters to connect modules and components to each other.
  e) Generation of derived components, is used when there is a higher level language that can be used for a particular task, which is then used to create the actual component.

– **C.1.2.2 Mechanisms for Achieving Variability**
  Domain engineering area should be supported with a knowledge about different variability mechanisms to make it possible to choose the most suitable one for concrete situations.

* C.1.2.2.ea1 Using Application Specific Plug-ins **SPLEMM - 2**
  In cases where variability provided by domain architecture is not sufficient to application needs, then a ad hoc solution must be developed. Although this is done by application developer, also domain architect must have prepared the architecture for such application-specific variants. Proper planning and trade-off decisions have to be made to have this process go easyest way possible.

* C.1.2.2.ea2 Definition of Configuration System **SPLEMM - 2**
  Different configuration methods (eg. license-key-driven configuration) exist to define system structure at build time. Foremost it is crucial to properly document usable parameters in configuration files to guarantee proper usage of those.

* C.1.2.2.ea3 Using Language and Generative Support **SPLEMM - 2**
  Methodologies in this group include aspect oriented program-

ming, template meta programming, domain specific languages and code generation and macro languages that allow compile-time binding.

- **C.1.2.3 Documentation of Variability**
  There is a lack of proven and available variability documentation mechanisms. Organisation wide methodology has to be agreed for it and its usage enforced.

  * C.1.2.3.a1 Using Orthogonal Variability Model **SPLEMM - 2**
    Orthogonal variability model is a method for variability documentation that allows descriptive models to be light-weight. The method documents only variable aspects of product line when some other methods grow fast in complexity as they capture both common and variable features.

- **C.1.2.4 Defined Levels of Variation Support**
  Product line can manage variability in different levels of complexity.

  * C.1.2.4.ea1 Managing Cross-Cutting Variants **SPLEMM - 1**
    Support for variation that are common for all applications of product line. Product line specifies only the variation points that exist over all the applications.

  * C.1.2.4.ea3 Introducing Variation Points **SPLEMM - 1**
    Variation points are defined in reference architecture. Through variation points it is determined how core assets can be configured to derive applications from reference architecture.

  * C.1.2.4.ea4 Completely Defined Variation Points **SPLEMM - 2**
    In more mature and planned ahead product lines there is no need for adding new variants to variation points during product derivation. Complete set of variants for each variation point are part of the configurable product base and ideally represented in derivation tool.

- C.1.2.ea1 Organising Variability on Various Levels of Product Line **SPLEMM - 3**
  Variability management is understood as process on different levels by addressing right issues and using proper variability techniques on each level of variability[29]:
  a) On *product line level* it is concerned how products differ over the product line. Variability management activities on this level include suitable components for product and generate or select product specific code from reference repository. b) On *product level* components are to be fit together and the parts of product specific code are replaced or extracted. c) On *component level* variability defines how to add implementations of the component interface, and also how these evolve over time. Questions addressed here include how to enable

addition and usage of several component implementations and how to design the component interface in such a way that it survives the addition of more concrete implementations. d) On *sub-component level* a component consists of a number of feature sets. On the sub-component level these feature sets are selected to create the component for a particular product Main issue on this level is removing or adding parts of a component where each part spans all component implementations. e) On *code level* evolution, but also most variability between products, actually take place. It has to be guaranteed that the provided class interfaces match the method calls performed throughout evolution of components.

C.2 - Configuration Management

Establish and maintain the integrity of all the work products of a process or project by managing variability in both time and space.
[1, 3, 27, 17, 2, 23]

- **C.2.1 Defined Configuration Management Process and Strategy**
  Determine configuration management strategy, including configuration management activities and schedule for performing these activities. Configuration management in SPLE context has to support few aspects that are not present in single system development environments.

  - C.2.1.ea1 Maintaining Unified Process of Configuration Management **SPLEMM - 2**
    In product line configuration management, a configuration must be maintained for each version *of each product* by a single process. Thus the process has one level of extra complexity when compared to configuration management in single system development.
  - C.2.1.ea2 Manage Permissions for Core Asset Product Developers **SPLEMM - 2**
    In SPLE context are core assets produced by one team but often used by one or several other teams. This sets the higher requirements for permission management complexity that single system development.

  - **C.2.1.1 Tool and Process Support Level**
    Depending on a complexity of variability management in organisation the needs for tool support differ significantly. It is important to match the needs with appropriate level of tool support.

    * C.2.1.1.ea1 Possiblility to Add New Application in Separate Branch **SPLEMM - 1**
      In case of adding new projects to product line it might be feasible to first create a separate branch and to integrate the changes into the main branch later. Configuration management tools and processes should allow this option.

* C.2.1.1.ea2 Basic Configuration Management **SPLEMM - 1**
  Configuration management system supports version, branch, baseline and branched baseline management. This functionality is supported in all commercially available configuration management systems.

* C.2.1.1.ea3 Component Composition **SPLEMM - 2**
  Composition and branched composition management where configuration management system is able to take snapshots of consistent compositions of component versions and maintain independent branches of component compositions. This is supported by few commercially available configuration management systems or homegrown systems are used.

* C.2.1.1.ea4 Software Mass Customisation **SPLEMM - 3**
  In this level configuration management has to handle a) variation point management by maintaining variants of files in domain space, b) customisation management by managing consistent compositions of common and variant files and b) Customisation composition management by managing compositions of customised components.

C.3 Organisation

Organisational issues relevant to SMEs in SPLE environment
[3, 10, 12, 9, 30, 23, 32]

- **C.3.1 Structure**
  Organisation structure that puts the roles and responsibilities into practice.

  - **C.3.1.1 Decision of Organisational Structure**
    SPLE and company specific aspects should be considered when deciding about the structural approach. Primary question is whether to have separate core asset groups or not.

    * C.3.1.1.ea1 Consider the Size of the Effort and the Number of Products **SPLEMM - 1**
      Usual practice is that with more complex product lines consisting of different systems and product groups and involving many developers, it is more beneficial to have dedicated core asset team as otherwise the communication processes would grow too complex.

    * C.3.1.1.ea2 Consider the High or Low Effort of Tailoring Core Assets **SPLEMM - 1**
      How much work does it take to tailor core assets into end products? If there is more effort in product tailoring compared to core assets development, then it could be beneficial to have integrated product groups with core asset responsibilities divided

between teams.

* C.3.1.1.ea3 Consider Proportion of New Development **SPLEMM - 2**
  Does the building of product line mean mainly new development or more legacy-based development? In latter case it is better to have product developers, who have experience with legacy system, to be responsible for mining legacy system and developing core asset base.

* C.3.1.1.ea4 Consider the Funding Model **SPLEMM - 2**
  There might be problems with funding core asset team as it is not clear who pays for their work. This issue must be cleared before domain engineering team is created.

* C.3.1.1.ea5 Consider the Volatility of Core Assets **SPLEMM - 2**
  If core assets are changing and evolving much they need more attention. Then it should be considered to have a dedicated group responsible for core asset management.

* C.3.1.1.ea6 Consider Parallel or Sequential Product Development **SPLEMM - 2**
  Whether the applications are built sequentially or in parallel has role in deciding about the organisational structure. In case of parallel production the need for separate core asset team is greater in order to avoid a chance that parallel teams are developing same functionality.

– **C.3.1.2 Organisational Structure Definition**
  Different structural solutions have been tested in industry. Choice of the best suitable solution depends on organisation specific issues. Different possible solutions should be considered before deciding on, and defining the final structure.

  * C.3.1.2.ea1 Business units specialised around types of products **SPLEMM - 1**
    The structure is organised around project based single system development. There exists no separate core assets team.

  * C.3.1.2.ea2 Single Development Department **SPLEMM - 1**
    There is one development team handling both reference architecture and product derivation from it.

  * C.3.1.2.ea3 Temporary Domain Engineering Units for Domain Development **SPLEMM - 2**
    Domain development unit is extracted from single development department on bases of need.

* C.3.1.2.ea4 Project Based Domain Engineering **SPLEMM -2**
  Domain engineering unit responsible for the design, development, and evolution of reusable assets. On less mature organisations the structure is still focussed on doing projects and certain senior resources are allocated to reusable component identification and development. In more mature organisations the domain and application roles are distributed over the organisation. There are separate domain engineering projects. Both domain and application engineering have mostly project-oriented structure. Most of the case studies have avoided the restructuring costs even if it holds down some of the benefits of SPLE. Decision of having separate domain engineering units has to be considered strongly before.

* C.3.1.2.ea5 Definition of Feedback Loops **SPLEMM - 2**
  Initiation of feedback loops between different product line related functional units (eg. application and platform developers) should be defined in organisational structure to guarantee the full functianality of them.

- **C.3.2 Roles and Responsibilities**
  Organisation has to manage the distinct responsibilities and relationships occurring in the software product family engineering.

  – **C.3.2.1 Definitions of Roles and Responsibilities**
    Distinctive domain and application engineering roles are defined in role definitions.

    * C.3.2.1.ea1 Ad Hoc Collaboration for Common Asset Development **SPLEMM - 1**
      There are no explicitly defined domain-engineering roles, but the application-engineering experts collaborate over project borders to identify and share common assets.

    * C.3.2.1.ea2 Emphasise Skills and Education **SPLEMM - 1**
      Both SPLE and SME environment sets higher requirements on level of employers skills and knowledge. SPLE has more complicated processes and more variables that employers have to be aware of. Also staff requirements are very high on the design of reusable assets and should be addressed that way. SMEs relay heavily on their employees knowledge and experiences to stay competitive. Recruitment and resource management should consider this issue.

    * C.3.2.1.ea3 Defining Both Application and Domain Engineering Roles **SPLEMM - 2**
      Both domain and application engineering roles and responsibilities are defined. Following responsibilities could be considered

: product manager, scoping team, domain expert, architecture manager, change manager, component developer, issue tracker, request dispatcher, configuration manager, build manager, and test engineer.

* C.3.2.1.ea4 Integrating Roles Between Domain and Application Engineering **SPLEMM - 2**
  There are coordination roles between domain and application engineering, and across domain-engineering organisations. Domain engineering has a major role in software development. In organisation with very high maturity domain and application engineering are integrated. The most important roles are the domain-engineering roles. Most people are involved in both domain and application engineering roles.

* C.3.2.1.ea5 Assignment of Basic SPLE Specific Tasks **SPLEMM - 1**
  Organisation should assign SPLE specific tasks to concrete units. There are several tasks that should have responsibles in the structure or its abbsence should be reasoned. These tasks include responsibles for production strategy, product scope and associated business case, product line architecture, requirements of product line and its members, designing and producing core assets, product deriviation.

* C.3.2.1.ea6 Assignment of Advanced SPLE Specific Tasks **SPLEMM - 2**
  Additional responsibilities to think about are core asset evolution, process improvement, production environment, forecasting and communicating new trends that can affect product line. Also the roles of produt, product line and domain and component architect could be divided if resources allow that.

– **C.3.3 Collaboration Processes**
  SPLE organisation may be uncommon and confusing to single system developers in the beginning. Thus some guiding aids should be used to lower the adaption parrier and enforce the usage of SPLE practices.

* C.3.3.1 Product Line Texture
  Texture is a collection of rules for coordinatint the implementation of the architecture and evolving it over time. These rules can be in form of coding conventions, design patterns (factory, strategy, extension interface, bridge and adapter), architecture styles (event based communication, pipes and filters, layered arch), or even usage of concrete framework. The rules has to be communicated effectively throughout organisation. Active awareness creation and dissemination of SPL principles are critical for success.

- C.3.3.1.ea1 Component Variability Documentation **SPLEMM - 1**
  Each core component must have associated process defined so that built in component variations can be exercised in application derivation.

- C.3.3.1.ea2 Architectural Constraints **SPLEMM - 2**
  Architectural constraints are used to restrict reuse to common platform

- C.3.3.1.ea3 Appropriate Organisation of Core Asset Base **SPLEMM - 2**
  Core asset base should be organised in a way that supports the easy access and usage of reusable components. Depending on exact context choice may be made among several options like key domain abstraction, architecture based, or feature based approaches.

- C.3.3.1.ea4 Rigidly defined variation points **SPLEMM - 3**
  There is an explicit reference architecture determining explicitly where application architectures may vary. Managed use of variation points supports the quality in whole SPLE processes.

- C.3.3.1.ea5 Product derivation **SPLEMM - 3**
  Automated product derivation mechanisms that support the correct use of product line architecture

5 Example Action Summary

Example Actions SPLEMM Level 1

Business MA

EA id	EA name
	Sales and Marketing
B.1.1.ea1	Product Line Wide Brand Name
B.1.1.ea2	Defined Branding Strategy
B.1.2.ea1	Using Process Qualities in Market Orientation
	Scoping
B.2.1.1.ea1	Studying Available Products in Existing Product Lines
B.2.1.1.ea2	Getting Information from Stakeholders
B.2.1.2.ea2	Define Domain Candidates
B.2.1.2.ea1	Specifying Product-Feature Matrix
B.2.1.3.ea1	Domain analysis
B.2.1.3.ea2	Identification and Prioritization of Product Line Goals
B.2.2.ea1	Definition of Scope
	Business Planning
B.3.1.2.ea1	Using Vision as Decision Making Tool

Domain Engineering MA

EA id	EA name
	Domain Requirements Engineering
D.1.1.2.ea1	Prioritisation of Requirements
D.1.1.2.ea2	Separation of Problem and Solution Space
D.1.2.1.ea1	Application-Requirements Matrix
D.1.2.2.ea1	Defining Variation Points and Variants in Requirements
D.1.2.2.ea2	Distinguish Between Internal and External Variability
D.1.3.1.ea1	Communicate Software Requirements
D.1.3.1.ea4	Reactive Customisation of Product Line Requirements
D.1.3.1.ea1	People Carrying Knowledge
	Domain Definition
D.2.2.ea1	Interface Design
D.2.2.ea2	Interface Implementation
D.2.3.1.ea1	Separation of the Common and Variant Behaviour
D.2.3.3.ea1	Defined Standard Activity
D.2.5.2.ea1	Ad-hoc Refactoring
	Domain Testing
D.3.1.1.ea1	Using a Sample Application Strategy (SAS)
D.3.1.1.ea2	Using Commonality and Reuse Strategy (CRS)
D.3.1.1.ea3	Composite Strategy
D.3.2.1.ea1	Creating Logical Test Cases
D.3.2.1.ea2	Creating Detailed Test Cases
	Product Line Architecture Documentation
D.4.1.ea1	Assigning Role for Architecture Documentation Responsible
D.4.1.ea2	Documentation and Guidelines for Application Builders
D.4.1.ea3	Communicate Architecture
D.4.2.ea1	Using Architectural Scenarios

Application Engineering MA

EA id	EA name
	Application Requirements Engineering
A.1.1.1.ea1	Analysing Target Group Needs
A.1.1.2.ea1	Communication of External Variability to Stakeholders
A.1.1.2.ea2	Usage of Domain Variability Model
A.1.2.1.ea1	Review Application Specific Requirements
A.1.2.3.ea1	Comparing Application Requirements to Product Line Scope Definition
	Application Design
A.2.1.ea1	Establishing Application Specific Design Documentation
A.2.2.1.ea1	Utilisation of the Organisation of Core Asset Base
A.2.3.ea1	Selecting Suitable Variants in Variation Points
A.2.3.ea2	Binding of Variants
	Application Realisation
A.3.ea1	Configuration
A.3.ea2	Realisation of Application-Specific Components
A.3.ea3	Building the Application
	Application Testing
A.4.ea1	Handling SPLE Specific Aspects in Testing

Collaboration MA

EA id	EA name
	Variability Management
C.1.2.1.ea1	Choice of Binding Time for Variability Implementation
C.1.2.1.ea2	Choice of Variability Mechanisms
C.1.2.4.ea1	Managing Cross-Cutting Variants
C.1.2.4.ea3	Introducing Variation Points
	Configuration Management
C.2.1.1.ea1	Possiblility to Add New Application in Separate Branch
C.2.1.1.ea2	Basic Configuration Management
	Organisation
C.3.1.1.ea1	Consider the Size of the Effort and the Number of Products
C.3.1.1.ea2	Consider the High or Low Effort of Tailoring Core Assets
C.3.1.2.ea1	Business units specialised around types of products
C.3.1.2.ea2	Single Development Department
C.3.2.1.ea1	Ad Hoc Collaboration for Common Asset Development
C.3.2.1.ea2	Emphasise Skills and Education
C.3.2.1.ea5	Assignment of Basic SPLE Specific Tasks
C.3.3.1.ea1	Component Variability Documentation

Example Actions SPLEMM Level 2

Business MA

EA id	EA name
	Sales and Marketing
B.1.2.ea2	Using Product Qualities in Marketing
B.1.2.ea3	Defined Product Definition Strategy
B.1.2.ea4	Setting Pricing Policy with Respect to SPLE Specific Aspects
B.1.3.ea1	Centralised Customer Support
B.1.3.ea2	Managing Customer Interface
B.1.3.ea3	Collecting Feedback from Customers
B.1.3.ea4	Limiting Customer Choices with External Variability
B.1.3.ea5	Active Involvement of Customers
	Scoping
B.2.1.1.ea3	Using Information Sources Available in Organisation
B.2.1.3.ea3	Technology Forecasting
B.2.1.3.ea4	Composing Initial Sketch of Business Case
B.2.1.3.ea5	Using Domain Experts
B.2.2.2.1.ea1	Developing Product Line Scenarios
B.2.2.2.1.ea2	Defined Scope Management Processes
	Business Planning
B.3.1.1.ea1	Analiese of Internal and External Environment, Risks and Opportunities
B.3.1.1.ea2	Using Product Roadmaps
B.3.1.1.ea3	Distinguishing Between Domain and Application Planning
B.3.1.2.ea2	Incorporating SPLE in business vision
B.3.2.1.ea1	Establishing Budget for Domain Engineering
B.3.2.2.ea1	Prediction of Future Costs and Benefits Using Current Development Approach

Domain Engineering MA

EA id	EA name
	Domain Requirements Engineering
D.1.1.1.ea1	Involving Different Roles
D.1.1.1.ea2	Involving Different Sources
D.1.1.2.ea3	Defining Requirements Organisation Process
D.1.1.1.ea3	Studying Existing Applications
D.1.1.2.ea4	Categorise Requirements
D.1.2.1.ea2	Priority-Based Analysis Scheme
D.1.2.1.ea3	Check-List Based Analysis
D.1.2.2.ea3	Document the Common Requirements in Detail
D.1.3.1.ea2	Process for Integrating New Requirements
D.1.3.1.ea3	Receiving Feedback from Application Engineering Cycles
D.1.3.2.ea2	Definition of Requirements Management Process
D.1.3.2.ea3	Document Assumptions of Commonalities
D.1.3.2.ea4	Compromise Between Quality and Generality
D.1.3.2.ea5	Regular Roundtable Meetings
	Domain Definition
D.2.1.ea1	Defining Production Strategy
D.2.1.2.ea1	Aspect-oriented Software Development (AOSD)
D.2.1.2.ea2	Usage of Layered Architecture Style and Strategy Pattern
D.2.3.3.ea2	Breath-First Analysis of Options
D.2.4.1.ea1	Usage of Agile Principles
D.2.4.1.ea2	Usage of Test-Driven Development
D.2.4.1.ea3	Encapsulation of Legacy Systems
D.2.4.5.ea1	Automating Compilation Activities
D.2.5.1.ea1	Architecture design reviews
D.2.5.2.ea2	Including Refactoring Activities in Processes
D.2.5.2.ea3	Removing Unused Component Parts
D.2.5.2.ea4	Finding Emerging Abstractions
D.2.5.2.ea5	Incorporating Business Unit Architectures
D.2.5.2.ea6	Minimising Interfaces
	Domain Testing
D.3.2.1.ea3	Creating Traceability Links
D.3.3.ea1	Domain Unit Testing
D.3.3.ea2	Integration Testing on Domain Engineering Leve
D.3.3.ea3	System Testing on Domain Engineering Level
D.3.3.4	Documenting Test Cases
	Product Line Architecture Documentation
D.4.2.ea2	Using Different Views to Represent Architecture
D.4.2.ea3	Specifying Component Interfaces

Application Engineerign MA

EA id	EA name
	Application Requirements Engineering
A.1.1.1.ea2	Using Subject Matter Experts
A.1.2.1.ea2	Defined Decision Criteria Exists for Domain Wide Requirements
A.1.2.2.ea2	Analysing Requirement Deltas
A.1.2.2.ea1	Deciding about Requirement Deltas
A.1.2.3.ea2	Updating Product Line Scope Definition
A.1.3.ea1	Establishment of Traceability Links
A.1.3.ea2	Documentation of Variability Bindings
	Application Design
A.2.1.ea2	Product Line Principles are Used in Application Design
A.2.2.2.ea1	Limiting Further Development of Core-Asset Based Products
A.2.4.ea1	Integrating Application Specific Elements to Reference Architecture
A.2.4.ea2	Updating Application Design with Reusable Components
	Application Testing
A.4.ea2	Domain Test Artefact Reuse

Collaboration MA

EA id	EA name
	Variability Management
C.1.1.ea1	Incorporating Ongoing Variability Points Identification in Processes
C.1.1.ea2	Evolution of Variability
C.1.2.2.ea1	Using Application Specific Plug-ins
C.1.2.2.ea2	Definition of Configuration System
C.1.2.2.ea3	Using Language and Generative Support
C.1.2.3.a1	Using Orthogonal Variability Model
C.1.2.4.ea4	Completely Defined Variation Points
	Configuration Management
C.2.1.ea1	Maintaining Unified Process of Configuration Management
C.2.1.ea2	Manage Permissions for Core Asset Product Developers
C.2.1.1.ea3	Component Composition
	Organisation
C.3.1.1.ea3	Consider Proportion of New Development
C.3.1.1.ea4	Consider the Funding Model
C.3.1.1.ea5	Consider the Volatility of Core Assets
C.3.1.1.ea6	Consider Parallel or Sequential Product Development
C.3.1.2.ea3	Temporary Domain Engineering Units for Domain Development
C.3.1.2.ea4	Project Based Domain Engineering
C.3.1.2.ea5	Definition of Feedback Loops
C.3.2.1.ea3	Defining Both Application and Domain Engineering Roles
C.3.2.1.ea4	Integrating Roles Between Domain and Application Engineering
C.3.2.1.ea6	Assignment of Advanced SPLE Specific Tasks
C.3.3.1.ea2	Architectural Constraints
C.3.3.1.ea3	Appropriate Organisation of Core Asset Base

Example Actions SPLEMM Level 3

Business MA

EA id	EA name
	Sales and Marketing
B.1.2.ea5	Strategical Order of Entry Decisions
	Scoping
B.2.1.3.ea6	Using Formal Methods for Domain Analyses
B.2.2.2.1.ea3	Adopting Comprehensive Scoping Process
	Business Planning
B.3.1.1.ea4	Defined Portfolio Management Processes
B.3.1.2.ea2	Quantitatively Incorporating SPLE in Business Vision
B.3.2.1.ea2	Managing Domain Engineering Budget
B.3.2.2.ea2	Prediction of Future Cost and Benefits of Product Line Approach
B.3.2.2.ea3	Investment Analysis about Product Line Related Costs

Domain Engineering MA

EA id	EA name
	Domain Requirements Engineering
D.1.1.2.ea5	Using Proven Representation Methods
D.1.3.2.ea6	Usage of Defined Requirement Management Tools or Processes
	Domain Definition
D.2.3.3.ea3	Using More Sophisticated Techniques and Tools
D.2.5.1.ea2	formal architecture evaluation methods
D.2.5.1.ea3	Establishing Arhitecture Review Board

Collaboration MA

EA id	EA name
	Variability Management
C.1.2.ea1	Organising Variability on Various Levels of Product Line
	Configuration Management
C.2.1.1.ea4	Software Mass Customisation
	Organisation
C.3.3.1.ea4	Rigidly defined variation points
C.3.3.1.ea5	Product derivation

References

[1] Spice assessment model.

[2] *Software Product Lines*, chapter Variation Management for Software Production Lines. Springer, 2002.

[3] Software engineering institute's website on software product lines. http://www.sei.cmu.edu/productlines/, May 2009.

[4] F. Ahmed, L.F. Capretz, and A. Jaffar. The business of software product family: An empirical survey. pages 745–749, 29 2008-Sept. 2 2008.

[5] Faheem Ahmed and Luiz Fernando Capretz. Managing the business of software product line: An empirical investigation of key business factors. *Information and Software Technology*, 49(2):194 – 208, 2007.

[6] Faheem Ahmed and Luiz Fernando Capretz. The software product line architecture: An empirical investigation of key process activities. *Information and Software Technology*, 50(11):1098 – 1113, 2008.

[7] 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. In *SSR '99: Proceedings of the 1999 symposium on Software reusability*, pages 122–131, New York, NY, USA, 1999. ACM.

[8] D. Beuche, A. Birk, H. Dreier, A. Fleischmann, H. Galle, G. Heller, D. Janzen, I. John, R.T. Kolagari, T. von der Massen, and A. Wolfram. Using requirements management tools in software product line engineering: The state of the practice. pages 84–96, Sept. 2007.

[9] Andreas Birk, Gerald Heller, Isabel John, Klaus Schmid, Thomas von der Maen, and Klaus Mller. Product line engineering: The state of the practice. *IEEE Software*, 20(6):52–60, 2003.

[10] Jan Bosch. Product-line architectures in industry: a case study. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 544–554, New York, NY, USA, 1999. ACM.

[11] Jan Bosch. *Design and use of software architectures: adopting and evolving a product-line approach*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[12] Jan Bosch. Maturity and evolution in software product lines: Approaches, artefacts and organization. In *Software Product Lines*, volume 2379/2002, pages 247–262. Springer, Berlin / Heidelberg, 2002.

[13] R. Carbon, J. Knodel, D. Muthig, and G. Meier. Providing feedback from application to family engineering - the product line planning game at the testo ag. pages 180–189, Sept. 2008.

[14] D. Ganesan, J. Knodel, R. Kolb, U. Haury, and G. Meier. Comparing costs and benefits of different test strategies for a software product line: A study from testo ag. pages 74–83, Sept. 2007.

[15] J.F. Girard, M. Verlage, and D. Ganesan. Monitoring the evolution of an oo system with metrics: an experience from the stock market software domain. pages 360–367, Sept. 2004.

[16] J.M. Hunt. Organizing the asset base for product derivation. pages 65–74, 2006.

[17] Danny Greefhorst Juha Kuusela J. Henk Obbink Jan Bosch, Gert Florijn and Klaus Pohl. Variability issues in software product lines. In *Software Product-Family Engineering*.

[18] I. John, J. Knodel, T. Lehner, and D. Muthig. A practical guide to product line scoping. pages 3–12, 0-0 2006.

[19] Michael Kircher, Christa Schwanninger, and Iris Groher. Transition to a software product family approach - challenges and best practices. In *Proceedings of the 10th International on Software Product Line Conference: IEEE Computer Society*, 2006.

[20] P. Knauber, D. Muthig, K. Schmid, and T. Wide. Applying product line concepts in small and medium-sized companies. *Software, IEEE*, 17(5):88–95, Sep/Oct 2000.

[21] Charles W. Krueger. New methods in software product line development. In *Proceedings of the 10th International Software Product Line Conference (SPLC 2006)*.

[22] John MacGregor. Requirements engineering in industrial product lines. In *Proceedings of the International Workshop on Requirements Engineering for Product Lines*.

[23] K Pohl, G Böckle, and F van der Linden. *Software product Line Engineering - Foundations, Principles, and Techniques*. Springer, 2005.

[24] R. Rabiser, P. Grunbacher, and D. Dhungana. Supporting product derivation by adapting and augmenting variability models. pages 141–150, Sept. 2007.

[25] Dale Churchett Ross Buhrdorf and Charles W. Krueger. Salions experience with a reactive software product line approach. In *Software Product-Family Engineering*.

[26] K. Schmid, K. Krennrich, and M. Eisenbarth. Requirements management for product lines: extending professional tools. pages 10 pp.–122, 0-0 2006.

[27] Software Engineering Institute. *CMMI for Development, version 1.2*, 2006.

[28] Thomas Fischer Andreas Hein Michael Schlick Steffen Thiel, Stefan Ferber. A case study in applying a product line approach for car periphery supervision systems. In *Proceedings of In-Vehicle Software*.

[29] Mikael Svahnberg and Jan Bosch. *Software Architectures for Product Families*, chapter Issues Concerning Variability in Software Product Lines, pages 146–157. Lecture Notes in Computer Science. Springer, Berlin / Heidelberg, 2000.

[30] Frank van der Linden. Family evaluation framework overview introduction. Technical report, ITEA project, 2005.

[31] Frank van der Linden, Klaus Schmid, and Eelco Rommes. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer, 2007.

[32] Martin Verlage and Thomas Kiesgen. Five years of product line engineering in a small company. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 534–543, New York, NY, USA, 2005. ACM.