Product Line Engineering for large-scale simulators
– An exploratory case study

En utforskande fallstudie av produktlinjer för utveckling av storskaliga simulatorprodukter

Felix Härnström

Supervisor : George Osipov
Examiner : Peter Fritzson

External supervisor : Johan Felixsson
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtidiga ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Felix Härnström
Abstract

This thesis takes a process-centric approach to Product Line Engineering (PLE) with the purpose of evaluating the suitability of PLE practices and processes in the context of large-scale industrial simulator products. This human-centered approach sets itself apart from previous research on the subject which has been mostly focused on architectural and technical aspects of PLE. The study took place at Saab, a Swedish aerospace and defense company whose primary product is the Saab 39 Gripen fighter aircraft. The study was conducted as a series of interviews with participants across three product lines, each responsible for a different line of simulators. By investigating their current working processes using the Family Evaluation Framework, a maturity rating was derived for each product line. This maturity rating was then considered alongside commonly reported issues and experiences in order to evaluate the usefulness of PLE practices for each product line. It was found that the studied organization could likely benefit from implementing PLE. PLE and the Family Evaluation Framework promotes practices that would alleviate some of the major issues found in the studied organization such as unclear requirements, issues with product integration and external dependencies, and a lack of quantitative data. Due to the relative immaturity of PLE processes in the studied organization, these conclusions are based on a review of existing literature and the stated goals and practices of PLE applied to the context of the studied organization.
Acknowledgments

I would like to extend my heartfelt thanks to the wonderful people at Saab that made this thesis possible. Particular thanks goes to my supervisor Johan Felixsson for supporting me the entire way, and to Robert Lindohf for his expert advice.

Furthermore I would like to thank my supervisor George Osipov and examiner Peter Fritzson at Linköping University for guiding me through the process of producing this thesis and providing valuable advice and insight.

Finally I would like to thank my family and my fiancée for their love and support throughout my studies.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Aim</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Research question</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Delimitations</td>
<td>2</td>
</tr>
<tr>
<td>2 Theory</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Product Line Engineering</td>
<td>3</td>
</tr>
<tr>
<td>2.1.1 Domain Engineering</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Application Engineering</td>
<td>7</td>
</tr>
<tr>
<td>2.1.3 PLE and agile development methods</td>
<td>9</td>
</tr>
<tr>
<td>2.1.4 PLE Adoption models</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Modeling and Simulation</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1 Systems Engineering</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Verification &amp; Validation</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Model-based Engineering</td>
<td>12</td>
</tr>
<tr>
<td>2.2.4 Combining PLE &amp; MBE</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Process Evaluation Frameworks</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1 CMMI</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2 The SEI Framework for Software Product Line Practice</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3 Family Evaluation Framework</td>
<td>17</td>
</tr>
<tr>
<td>2.3.4 SCAMPI</td>
<td>19</td>
</tr>
<tr>
<td>2.4 Methodology</td>
<td>20</td>
</tr>
<tr>
<td>2.4.1 Planning and conducting a case study</td>
<td>20</td>
</tr>
<tr>
<td>2.4.2 Interview techniques</td>
<td>22</td>
</tr>
<tr>
<td>2.4.3 Trustworthiness</td>
<td>22</td>
</tr>
<tr>
<td>3 Background</td>
<td>24</td>
</tr>
<tr>
<td>4 Method</td>
<td>26</td>
</tr>
<tr>
<td>4.1 Designing the case study</td>
<td>26</td>
</tr>
</tbody>
</table>
## List of Figures

2.1 The economics of PLE .................................................. 4
2.2 Typical PLE artifacts ................................................... 4
2.3 PLE sub-processes ...................................................... 7
2.4 Model cost versus value ............................................... 12
2.5 Staged and continuous representations of the CMMI framework .............................................. 14
2.6 Staged CMMI components ........................................... 15
2.7 BAPO model ............................................................. 17
2.8 Dimensions and levels of the FEF .................................. 18
4.1 Interview structure ..................................................... 28
4.2 Data coding and analysis .............................................. 29
List of Tables

5.1 Extracted codes ...................................................... 31
5.2 Product line maturity ratings .................................... 38
List of Abbreviations

**BAPO**  Business, Architecture, Process, Organization  
**CMMI**  Capability Maturity Model Integration  
**DARE**  Domain Analysis and Reuse Environment  
**FEF**  Family Evaluation Framework  
**FODA**  Feature-Oriented Domain Analysis  
**MBE**  Model-Based Engineering  
**M&S**  Modeling and Simulation  
**PLE**  Product Line Engineering  
**SCAMPI**  Standard CMMI Appraisal Method for Process Improvement  
**SPLE**  Software Product Line Engineering
Large-scale software and hardware development projects have seen a long-standing trend of increased complexity and a corresponding increase in development cost, time and effort. In response to this, industrial practitioners are increasingly adopting Product Line Engineering (PLE) in order to manage complexity and deliver higher-quality products at a lower cost and effort [35]. This shift calls for a holistic approach in software development to allow for increased reuse of previously developed assets when developing new products. In PLE, this applies to almost all development artifacts: requirements, designs, applications and tests should all be shared across products where appropriate. This thesis presents a case study of how PLE may be used to construct large-scale simulators in the aerospace and defense industry, with the particular objective of evaluating the suitability of PLE development processes in this context.

1.1 Motivation

It is clear that PLE is a compelling approach to developing large-scale modeling and simulation (M&S), similarly to how it has been used in other industries and applications. The virtues of PLE for construction of large-scale simulator products have been described by for instance Andersson, Herzog, and Ölvander [1], and Wittman Jr and Harrison [49]. PLE is already established as a reasonably mature approach to software development according to Arboleda and Royer [3] and seems to be a compelling and practical approach to managing complexity in simulation-heavy projects as shown by Andersson, Herzog, and Ölvander [1] and Wittman Jr and Harrison [49]. More mature approaches even take a model-driven approach to product lines themselves by merging PLE with model-based engineering (MBE) [3, 30].

These previous experiences of PLE and M&S show that it is clearly possible, and perhaps even suitable, to adopt a product-line approach to large-scale simulator products. However, the studies that exist on this subject are mostly focused towards the technical and architectural aspects of PLE. While this is indeed a fundamental concern in any attempt to construct a holistic PLE framework for large-scale simulators, it should arguably not be the only concern. In order to evaluate the actual day-to-day experience of working within such a framework we have to instead look at the human-centered development process that such a framework imposes. This thesis takes a process-centered approach to PLE for M&S as a complement to the already established views on the technical and architectural compatibility of these paradigms.
1.2 Aim

This thesis explores the practical suitability of developing large-scale simulator products as parts of a larger software product line, with a particular focus on development processes. In particular, the suitability of introducing product lines for preexisting simulator software components is evaluated. Such an evaluation takes into consideration the development process necessitated by PLE together with practices required for M&S, and evaluates the combination of working processes on a a basis of maturity and compatibility. This will complement previous studies on PLE for M&S by exploring whether the technical compatibility and individual advantages translate to a practically viable combination of development processes in the context of large-scale simulator products. This will be done as an in-depth case study on the use of PLE for construction of large-scale simulator products for the Saab 39 Gripen fighter aircraft.

1.3 Research question

Based on the stated aim of this thesis as elaborated upon in previous sections, a single main research question may be formulated:

To what extent are PLE development practices suitable for developing large-scale simulators?

The main research question will draw upon the findings of the following research objectives:

RO1 Determine what development processes are recommended or required (both explicitly and implicitly) for PLE and M&S, respectively.

RO2 Establish a framework for measuring process suitability and maturity.

These research objectives are intended to support the main research question and will serve as a guide for establishing a theoretical and analytical framework as well as the empirical approach.

1.4 Delimitations

As previously stated, this thesis is focused on process-centered aspects of PLE and M&S. This means that technical, architectural, business, and organizational-focused concerns of compatibility are not investigated. Furthermore, the practical case study is restricted to a single organization (Saab) in order to ensure a basic level of homogeneity between investigated product lines.
This chapter provides an overview of relevant theories and previous work on the topics of PLE and M&S. It also covers methods for evaluating process effectiveness and maturity in the context of large-scale software development. This provides the foundational theoretical framework for process evaluation as described in chapter 4 and applied in chapter 6.

2.1 Product Line Engineering

Software Product Line Engineering is at its foundation an engineering principle designed in the image of traditional factory manufacturing lines. It proposes that software can be built in a product line fashion from a common base platform with variations of that base platform that can be suited to the needs of different customers. This presents an alluring compromise for both producers and customers; by being able to reuse parts of the common base platform, the cost of development goes down while the ability to provide a versatile set of products is retained or even increased. Of course, this entails a higher up-front cost both in terms of money and effort, but studies have shown that the cumulative cost of product lines may undercut more traditional development methods after just three product development projects. The relationship between the cumulative cost of PLE versus traditional product development is visualized in figure 2.1. Aside from a reduction in development costs, Pohl, Böckle, and Linden argue that PLE also leads to an increase in product quality because software components that are reused across several products are more mature and thoroughly tested than one-off equivalents, and that the time to deploy new products to the market is reduced after the initial effort of setting up the product line. Linden, Schmid, and Rommes also highlight reduced maintenance costs and lower overall project risk due to the reduced scope of the related software development project. Of particular interest for the subject of this thesis, they also note that product lines are inherently well-suited to modeling and simulation of embedded systems since they can be constructed such that a product as well as its simulation are both treated as variants of the same product line.

Given this rationale for adopting a product line approach to software development, let us consider the overall structure and process of setting up a software product line. Linden, Schmid, and Rommes presents a framework for PLE that separates development activities and artifacts along two parallel areas of concern: domain engineering and application engineering. Domain engineering is the process of developing the shared assets and tools that make up
2.1. Product Line Engineering

Figure 2.1: The economics of PLE. Adapted from [6].

the product line infrastructure. Application engineering, on the other hand, is essentially the act of realizing an actual product from the shared PLE platform. This is succinctly expressed by Atkinson and Muthig: domain engineering is “Development for Reuse” ([5], p. 103) and application engineering is “Development for Use” ([5], p. 103). This divide between domain and application engineering is mirrored by Pohl, Böckle, and Linden [35] who add that both of these processes are concerned with essentially the same types of basic artifacts: requirements, architectures, source components, and tests. The difference lies in their abstraction level and purpose: while domain engineering is concerned with establishing a shared 'pool' of these artifacts as well as defining the commonality and variability of the product line as a whole, application engineering draws upon this platform to minimize development effort of individual products [35]. This relationship between domain and application engineering is visualized in figure 2.2.

Figure 2.2: Typical PLE artifacts. Adapted from [23].
2.1.1 Domain Engineering

As previously stated, domain engineering is essentially the practice of developing generic artifacts (architectures, software components etc.) with the express purpose of reuse in concrete products [5, 23, 35]. This chapter provides an overview of frameworks and processes that belong to the realm of domain engineering.

Draco

A systematic approach to domain engineering was formalized by Neighbors as the Draco framework [20, 31]. This approach describes domain engineering as the analysis and design of three types of fundamental domains, referred to as the Application-, Modeling-, and Execution domains [31]. These domains are hierarchical and descending in abstractness, with application domains setting constraints on the modeling domains, and modeling domains dictating the choice and design of execution domains [31]. Neighbors describes these domains as:

**Application domains** Act as a sort of glue between modeling domains and decides the overall system constraints. Application domains can treat modeling domains as abstract data processing units and construct a model for the overall system-level data flow without detailing how that data is processed.

**Modeling domains** A well-defined subset of objects, operations and translation rules that encapsulate a single major function in the system.

**Execution domains** The domain in which the system is realized.

DARE

Aside from defining these domains, Neighbors also highlights how the analysis and design of these domains must be grounded in several different views of the system: available technologies, user needs, preexisting systems and the possibility of reuse, only to name a few [31]. The importance of considering all these different aspects of the system and its environment is mirrored in the Domain Analysis and Reuse Environment (DARE) framework [14]. This approach to domain analysis also explicitly incorporates the concepts of *commonality* and *variability*. Commonalities are such elements and relationships that are reused for most systems in the domain, while variabilities are those unique to only a few specific systems [14].

FODA

The goal when designing a domain architecture is to maximize commonality while still allowing for variability when necessary [14]. A common way to model commonality and variability is with a feature-oriented approach [19, 21]. Feature-oriented domain analysis (FODA) may be used to express commonality and variability in terms of user-facing features. This approach covers both the actual, expected use of the system as well as user requirements, and is therefore necessarily at a suitably high level of abstraction for designing an application domain [19, 31]. The FODA method is composed of three discrete phases [21]:

**Context analysis** A candidate domain is explored in terms of its relationships to external domains. This step should clearly locate the proposed model in the hierarchy of existing models and describe how these models interact.

**Domain modeling** Defines which specific problems that the components of the domain addresses. This is typically in the form of features, documentation, and requirements.

**Architecture modeling** Establishes generic architectural models to be used for software components in the domain. These models typically map the problems identified in the domain model to generic software architectures.
2.1. Product Line Engineering

Feature-oriented PLE

Kang, Jaejoon Lee, and Donohoe [19] build on the FODA method to present a framework for feature-oriented PLE, which enhances our view of the domain engineering process with a business and marketing perspective. This aspect of domain engineering ties in with product management and ensures that engineering decisions are made in accordance with business goals and strategies [19, 28, 35]. Kang, Jaejoon Lee, and Donohoe [19] argue that since system features are driven by business and marketing decisions, those decisions necessarily influence the decisions made in the domain engineering process. Reflecting this view, they propose a framework for feature-oriented PLE where the fundamental aspects of a proposed product line is expressed as a feature model consisting of both functional and non-functional product features. From this feature model, a high-level conceptual architecture design may be constructed. This architectural model describes conceptual components and relationships between these, from which the conceptual architecture may be realized into specific components, architectures and patterns that can be used throughout the system. Throughout this entire process, the business and marketing objectives can be used as a guide for defining quality attributes, eliciting requirements, features, and design options.

Product management

The approach to feature-oriented PLE suggested by Kang, Jaejoon Lee, and Donohoe [19] clearly shows how business plans may be used to guide the design of product lines. A firm connection to the business goals simplifies engineering decisions by providing a clear strategic direction and has been empirically found to be a success factor in domain engineering [28]. For the purpose of this thesis, we will use the term product management for this aspect of domain engineering, which mirrors how Pohl, Böckle, and Linden [35] use the term. They put product management in special relation to domain requirements engineering by letting the product roadmap and list of existing artifacts influence the requirements engineering activity, and letting the insights gained during this phase influence the business plan in turn. As such, product management is an iterative process of making sure that the goals and scope of the product line are aligned with the business plan, and that the business plan is kept grounded in the reality of the development process. In effect, it is a product management decision to delineate product scopes and feature sets. These decisions can be expressed as feature profiles which are formal, machine-readable product feature configurations that can be used by some automated PLE configurator to assemble finished (or nearly finished) products from the pool of shared assets [23].

Summary

Domain engineering is the process of developing a collection of shared assets for reuse across products [5]. The artifacts produced during this process are typically shared requirements, generic architectures, source components and tests [35]. The Draco framework divides the domain engineering into three separate types of domains: application domains, modeling domains and execution domains [31]. This framework asserts a hierarchical view of these domains: application domains are concerned with high-level system behavior, modeling domains defines abstract components and data flow paths, and execution domains express implementation-level constraints [31]. The goal when analyzing and designing these domains is to maximize commonality while still retaining the capacity for variability when realizing actual products [14]. An appealing method for exploring commonality and variability is the feature-oriented approach, which defines high-level commonality in terms of user-facing features [13, 21]. This approach is intimately connected to the product management aspect of domain engineering, which is the overarching process of aligning the product line with business and marketing goals [35]. A clear connection to the "strategic direction of the business has been shown to be an important success factor when establishing a product line [28]. In more practical terms,
the business plan can be used to construct feature profiles which are the formal configurations used to realize actual products from the common set of shared PLE artifacts [23].

2.1.2 Application Engineering

Application engineering is the process of actually developing a product from the shared PLE assets with the goal to maximize reuse of common assets and to exploit both the commonality and the variability of the product line. Pohl, Böckle, and Linden [35] define four sub-processes in application engineering:

**Requirements engineering** Eliciting requirements from stakeholders and domain requirements to produce a concrete requirements specification.

**Design** Designing a specialized application architecture from the shared domain architectures.

**Realization** Constructing the product according to the architecture, using preexisting assets where possible.

**Testing** Comprehensive product testing using both domain-generic and application-specific test artifacts and specifications.

The remainder of this chapter explains specific practices and goals for these sub-processes in more detail, as they are described by Pohl, Böckle, and Linden [35]. The relationship between these sub-processes is illustrated in figure 2.3 and described in further detail in the remainder of this chapter.

![Figure 2.3: PLE sub-processes.](image)

Adapted from [35].

**Application Requirements Engineering**

Requirements engineering in the area of applications engineering is essentially the process of eliciting and applying requirements relevant for the product whilst maximizing reuse of domain requirements. This process is tightly connected with product management activities, domain requirements engineering, and application design.

The choices made in product management defines major features of the product line which must be reflected in the application-specific requirements, but insight gained during application requirements engineering must also propagate back to the overarching product management strategy. Such feedback may lead to changes in how product line artifacts are applied, or even to new artifacts being developed.
The connection between applications requirements engineering and domain requirements engineering comes via the reuse of domain requirements. As previously stated, the objective is to maximize the reuse of domain requirements and to only use application-specific requirements when the trade-off between engineering effort and delivered value is favorable. In certain situations it may even be the case that requirements elicited for a particular application are incorporated into the corpus of domain requirements.

Finally, application requirements engineering is a necessary prerequisite for application design, since the design must necessarily accommodate the requirements of the application. However, the design process may also lead to deeper understanding of the engineering challenges associated with the development of the application. This can in turn lead to modifications to existing requirements.

**Application Design**

The goal when designing a concrete application is to realize the product architecture from the available domain architectures and application requirements. The application architecture should be a specialization of the reference architecture realized through the binding of variants at predefined variation points in the reference architecture and the addition of application-specific design choices. Application design is related to application requirements engineering as previously described, but also to the domain design process and to application realization.

The influence of domain design on application design should be apparent through the reuse of domain architectures and selection of domain artifacts. However, it should be noted that application designs can also be brought up to the domain level if they are found to be useful for other products. It may also be the case that the context of a specific application necessitates changes to the overall domain designs as long as such changes may be made without disturbing products utilizing the same reference architectures.

How the application is realized is an activity on its own, but it should be noted that the design process does not necessarily end when the architecture is handed off to be realized. Indeed, realization of the application design often uncovers errors that must be corrected at the design level.

**Application Realization**

The application architecture is realized by configuring the specified domain artifacts and bringing them together with application-specific development artifacts. The goal of application realization is to produce the actual product that can then be tested and brought to market. This makes application realization highly dependent on the design artifacts created during application design, but also to the process of domain realization and to application testing.

Application realization relies on the work during domain realization by utilizing the domain artifacts and configuration tools to develop the shared parts of the product. The application-specific artifacts which are developed may also themselves be brought into the pool of domain artifacts.

Finally, application realization and application testing are dynamic and iterative processes. Realized artifacts and interfaces are tested as a separate activity, and the results of those tests can uncover defects which must be corrected.

**Application Testing**

The final application engineering activity described by Pohl, Böckle, and Linden is application testing. This is the final activity required to achieve sufficient product quality before the product is released to market. This activity covers unit testing, integration testing and system testing and is intended to be performed in close connection with all other application engineering activities.
Application testing utilizes the domain testing artifacts to reduce the application-specific testing effort. These domain artifacts are essentially reusable test artifacts that verify the generic domain requirements. This process is essentially a miniaturized process of application engineering, since application-specific tests must be constructed and configured in much the same way as other concrete products.

Other than this connection to the domain testing activity, application testing is closely connected to all other application engineering activities. Application testing involves finding defects in requirements and designs as well as the realized product. Similarly, the application-specific requirements and design artifacts must be considered when constructing tests, just as tests are designated around concrete applications and interfaces.

Taken together, application testing is arguably the most interconnected application engineering activity since it requires communication and sharing of artifacts across all other application engineering activities. The application testing activity covers both verification and validation of application engineering artifacts, and the test documentation is recorded as an application artifact in its own right.

### Summary

Application engineering is, as previously defined, “Development for Use” ([5], p. 103). This is the process of realizing concrete products from the shared set of domain artifacts together with application-specific components. Pohl, Böckle, and Linden ([35]) describes application engineering as an iterative process of four sub-activities: requirements engineering, design, realization and testing. The transition between these is essentially linear, but the overall process of application engineering is iterative at its core. The decisions made and artifacts produced in one activity are generally transferred to the next activity in line and also given as feedback to the previous activity. In this fashion, the process of application engineering as described by Pohl, Böckle, and Linden ([35]) manages to both realize concrete products and reinforce the shared set of domain artifacts with the lessons learned and artifacts produced for specific applications.

#### 2.1.3 PLE and agile development methods

Previous research has shown that PLE can be used effectively in conjunction with agile development methods such as Scrum. According to Díaz, Pérez, Alarcón, and Garbajosa ([13]), agile methods can be an effective method for dealing with unplanned change in organizations that use PLE. This is motivated by the planned nature of PLE; in theory, PLE provides the most benefits when the entire product line can be planned from the very start ([14]). However, since this is rarely feasible in practice (and even well-laid plans will quite certainly be deviated from), Díaz, Pérez, Alarcón, and Garbajosa ([13]) suggest that agile methods can be used deal with unplanned changes to software product lines for both domain and application engineering. Similar conclusions were drawn by O’Leary, McCaffery, Thiel, and Richardson ([33]), who found that agile methods may be used for product derivation in product line environment. For large organizations they argue that their agile approach to product derivation can provide a good “balance between formalism and agility” ([33], p. 568).

#### 2.1.4 PLE Adoption models

As previously described, PLE typically requires a rather substantial up-front investment in order to generate savings further down the line ([6, 29]). According to Krueger ([22]), even though PLE may provide substantial benefits, the costs, risks and organizational re-structuring required are a major barrier to entry for many organizations. Because different organizations have different needs and constraints on their adoption of PLE, several different adoption models have been developed in order to provide easier methods for adoption based on the organization’s needs. Krueger ([22]) presents three broad adoption models:
Proactive The proactive approach to PLE is analogous to the big-bang approach described by Schmid and Verlage [44]. This approach is essentially a green-field implementation of PLE where a product line is developed from scratch in order to support the organizations foreseeable needs. While such an approach is optimal in theory, it is rarely as easy to implement in practice [44].

Reactive Usually quicker than the proactive approach, the reactive approach sees the product line expanded continually as new business needs arise. This corresponds to the incremental approach described by Schmid and Verlage [44].

Extractive The extractive approach might be suitable where already existing components and systems can be reused. This approach essentially extracts the common aspects of the existing systems and constructs a product line around these elements. As such, the extractive approach may be suitable for organizations with an existing high level of commonality between products.

While the effort required to adopt PLE depends heavily on organization-specific factors, Schmid and Verlage [44] show that the proper use of available tools can reduce adoption efforts for all of the described adoption methods.

2.2 Modeling and Simulation

In the most basic sense of the term, a model is a stand-in for some real system which, when subjected to some kind of experiment, allows us to draw conclusions about the behavior of the real system that the model depicts. Depending on the context it may be worthwhile to reason about different types of models. Fritzson [15] gives examples of four basic types of models: mental models, verbal models, physical models and mathematical models. In the field of software development, verification and validation, we are likely most concerned with mathematical models. These types of models are used to describe a system in terms of its inputs and outputs, and mathematical formulas that govern the relationship between these variables [15]. Such a model is also particularly well-suited for computer-aided simulation, which can be critical tool in the development of large-scale cyber-physical systems.

2.2.1 Systems Engineering

Systems engineering as a concept stems from the need to manage complexity in large development and management projects and encourages a holistic approach to several development processes: planning, analysis, optimization, integration and evaluation [43]. Systems engineering acknowledges that the complexity of an entire system is greater than the sum of its parts, and highlights the importance of cross-disciplinary approaches in order to manage complexity [34].

Intimately connected to systems engineering is the concept of systems thinking. This is an all-encompassing approach to complex systems engineering that incorporates the engineer and their view of the system as an added layer of complexity, among other things [36]. A general definition of systems thinking is given by Richmond as “[…] the art and science of making reliable inferences about behavior by developing an increasingly deep understanding of underlying structure” ([36], p. 139). This approach to systems thinking rests on three core skills [36]:

System as Cause Thinking The very structure of a complex system can be the cause of many issues, as opposed to the idea that such problems are external to the system.

Closed-loop Thinking Systems are composed of many ‘closed loops’. The various aspects of the system all influence (and are influenced by) each other.
Operational Thinking The way these closed loops interact can cause feedback loops that create a dynamic behavior in the system which can be difficult to understand and follow.

This rather generic definition of systems thinking is extended upon by Arnold and Wade [4] who add that conceptual models can be used to reduce complexity by essentially transforming or abstracting our view of a system, and that systems may be defined at different scales or as compositions of several smaller systems. Their study also found several other common aspects of systems thinking, but these are not necessarily relevant for the purposes of this discussion.

Based on these definitions of systems engineering and systems thinking, it is clear how M&S can be used as a tool to reduce complexity in large-scale systems engineering. Selic [45] suggests that the value of a model used for M&S can be derived from five key characteristics:

**Abstraction** A model is necessarily an abstraction of the system, it hides parts of the whole rather than adding them. By only showing the parts of the system we are interested in and hiding the rest behind layers of abstraction, we can draw conclusions about the essential behavior of select parts of the system.

**Understandability** A good model is more understandable than the system or the implementation being modeled by using human-friendly abstractions.

**Accuracy** The model must mimic the actual behavior of the system in whichever aspect it is being modeled.

**Predictiveness** Experiments performed on the model can tell us something new about the behavior of the system even in previously untested situations.

**Cost** A good model is less expensive to construct and analyze than the system itself.

While Selic uses these characteristics as a way to assess the quality of a model in and of itself, they also demonstrate how M&S may be used in systems engineering to reduce complexity in large-scale development projects.

2.2.2 Verification & Validation

While previous chapters have highlighted the utility of M&S in order to reduce and manage complexity, they have not covered a key point of concern: how do we know whether to trust what our models tell us? This is part of model accuracy that Selic [45] identified as a key characteristic of a good model. In order to answer this question, this chapter will cover the topic of verification and validation of models.

Sargent [40] provides helpful definitions and explanations of model verification and validation. According to Sargent, model verification is the process of “ensuring that the computer program of the computerized model and its implementation are correct” ([40], p. 12). Model validation on the other hand is the “substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model” ([40], p. 12). Model verification is thus the process of determining the accuracy of particular computerized implementation of a model with regards to the underlying mathematical model, and model validation is the process of determining whether the model is an acceptable stand-in for the actual system in a particular scenario. Sargent [40] highlights an important observation to make when assessing the accuracy of the model, which is the particular domain in which the model is intended to be valid. It is rare that a model is valid for all possible situations and scenarios - instead a model typically has a restricted scope of intended use, and the accuracy of the model is evaluated with those restraints in mind. However, even with this reduced scope the cost of model validation can be significant. The fundamental relation between cost and value of the model is expressed in figure 2.4.
2.2. Modeling and Simulation

2.2.3 Model-based Engineering

When modeling and simulation (M&S) are integral parts of the development of large-scale systems, it might be constructive to introduce the term model-based engineering (MBE). Czarnecki, Antkiewicz, Kim, Lau, and Pietroszek [11] describes how this approach to software engineering generally does away with traditional document-based development artifacts (requirements, feature sets, tests etc.) in favor of constructing holistic models in which information is encoded. These models are treated as core development artifacts, from which other assets such as program code may be derived. Schätz, Pretschner, Huber, and Philipps [42] takes this a step further by considering these models to be the central descriptions of processes as well as products. This approach posits that the model represents the abstract desired properties of a system and that it is possible to more-or-less automatically realize a product from the model.

2.2.4 Combining PLE & MBE

Previous studies have shown that there are several viable approaches to combining PLE and MBE. However, this does not necessarily imply that those approaches are accepted as canonical or standardized either in academic or industrial contexts [3]. For instance, one approach may treat models as shared PLE assets - in effect allowing for the construction of more complex M&S artifacts from a shared set of domain models. Others take a model-driven approach to the product line itself or to variability management. This chapter provides an overview of existing studies on different approaches to combining PLE and MBE.

Models as shared assets

A common, albeit perhaps somewhat unrefined, approach to PLE+MBE is to consider models to be part of the common pool of shared domain artifacts [50]. Young, Cheatwood, Peterson, Flores, and Clements [50] describe how this approach has been successfully implemented by organizations across several different industries. Common to these companies’ approach to PLE+MBE was that M&S was used extensively in parts of the development process and in order to manage the complexity of using several different models in several different contexts. The models themselves were treated as shared PLE assets that could be realized into product-specific model configurations in the same manner as any other product is realized in the product line. The authors describe how this approach was implemented (with some variations) by three different organizations in the defense and automotive industries: Raytheon, General Dynamics and General Motors [50]. All three companies are described as using feature-oriented PLE to configure model variability. This means that the feature model defines points of variability for a particular product based on the desired features of that product, which is then realized.

Figure 2.4: Model cost versus value. Source: [40].
via the reuse of shared PLE artifacts (and potentially product-specific assets). Extrapolating from this, since this approach treats models as shared domain artifacts, this approach can even be used to construct more complex models of models if an appropriate feature profile for the complex model is constructed.

**Feature-driven product lines**

A FODA-inspired approach to PLE+MBE could be described as a model-driven product line. A popular approach to this is - as touched upon by Young, Cheatwood, Peterson, Flores, and Clements \[50\] - feature based modeling of product lines. This is proposed by Czarnecki and Antkiewicz \[10\] with their approach to feature-based model templates. Their work presents a tool for feature-based inclusion or exclusion from a set of possible allowable configurations. This shows how feature-based modeling can be used as a practical tool to generate product-specific feature profiles which in turn may be used to derive product line artifacts.

This way of using feature models and points of variability as core assets in the product line is also proposed by Schätz \[41\]. Their approach similarly utilizes an underlying model for the generic system description, upon which a variability model is applied. In particular, their work takes a formal approach to modeling domain-specific variability, which the author claims sets it apart from other model-driven approaches to PLE. However, it still conforms to the core tenets of feature-oriented PLE: that a product line can be modeled in terms of available features and particular configurations of these features.

This feature-oriented approach to model-driven product lines is also described by Apel, Batory, Kästner, and Saake \[2\], who present a comprehensive approach to feature-oriented product lines. They also classify variability modeling as a domain engineering process, but puts it in direct relation to application requirements engineering. How variability is modeled directly affects how stakeholders’ requirements may be mapped to available features. As such, variability modeling needs to be closely connected to both the domain analysis as well as stakeholder requirements in order to be a useful tool for feature-driven product lines.

Taken all together, these approaches to model-driven product lines all echo the same core sentiment: that product lines can be modeled in terms of available features and variability between products. This is in line with the underlying idea of product lines as a mechanism for reuse. By expressing the available shared assets as features (or whatever that may translate to in a particular domain-specific context) and modeling the relationships and dependencies between these features, product configuration can also be tightly connected to requirements engineering. Such approaches both combine the aspect of reuse from PLE with the expressiveness of MBE, and provide convenient means of exchange between domain and application engineering.

### 2.3 Process Evaluation Frameworks

There exists several frameworks for evaluating the effectiveness and maturity of PLE processes. This chapter introduces the generic Capability Maturity Model Integration framework, followed by several PLE-specific frameworks.

#### 2.3.1 CMMI

The Capability Maturity Model Integration (CMMI) framework is a generic process improvement and assessment framework, designed to provide guidance for process improvement across several different process areas and organizational contexts \[9\]. The CMMI framework itself is designed around the concept of a CMMI model - a specific body of knowledge applicable to a specific discipline (e.g. engineering or software engineering) or a combination of disciplines. CMMI models are either *staged* or *continuous*. The difference between these two is how they approach the progression of process assessments and improvements, as illustrated in figure 2.5.
2.3. Process Evaluation Frameworks

(a) Staged CMMI representation

(b) Continuous CMMI representation

Figure 2.5: Staged and continuous representations of the CMMI framework.

Staged representation  The overall CMMI assessment is based on discrete maturity levels. Maturity levels require increasingly mature processes. The staged representation essentially treats process improvement as a series of steps, where the introduction of certain processes enables an organization to move to progressively higher rungs on the ladder.

Continuous representation  The overall assessment is organized around individual process areas, each with its own capability level. Improvement goals are defined for each process area, thus allowing for a customized progression path.

This thesis will focus on the staged representation of CMMI due to its use in the Family Evaluation Framework as described in chapter 2.3.3. However, both representations are designed to provide essentially the same results.

The staged representation defines a number of maturity levels which are essentially collections of processes. If the organization fulfills the goals of all process areas for a particular maturity level, the organization is considered to have reached that maturity level. There are five maturity levels:

1. **Initial**  Process are not formally managed and usually ad-hoc. Success depends on situational factors rather than established processes.

2. **Managed**  Requirements and processes are formally managed. Projects are performed according to established plans.

3. **Defined**  The organization maintains a standard set of processes which are improved over time. Project-specific processes are adapted from the set of standard processes. The standard set of processes is maintained proactively and are applied consistently throughout the organization.

4. **Quantitatively Managed**  Processes are selected, evaluated, and implemented using quantitative techniques. Metrics are collected and leveraged to support process decision making. Process performance can be predicted using quantitative techniques.

5. **Optimizing**  Processes are preemptively and continually optimized according to shifting business objectives. The responsibility for process improvement is shared throughout the entire organization.

Note that these are merely conceptual descriptions. The actual definition of a maturity level is given in terms of achieving its prescribed process areas and corresponding goals.
2.3. Process Evaluation Frameworks

CMMI process areas are shared between both the staged and continuous representations. A process area is a collection of related processes and practices that are intended to be implemented and improved as a group. Each process area is composed of a number of specific and generic goals. Generic goals are, as the name suggests, not specific to any particular process area, but are instead intended to enable the institutionalization of best practices in the organization. This is built a few key practices: adhering to policies, establishing formal plans and descriptions, providing resources and training, and evaluating and improving processes. Higher maturity levels have more demanding generic goals. Generic goals are organized around four common features: commitment to perform, ability to perform, directing implementation, and verifying implementation. In addition to the generic goals, CMMI process areas are composed of specific goals. These goals are tied to a particular process area and maturity level, and are intended for process assessment and improvement for that particular area.

Both generic and specific goals have certain practices attached to them. These are aptly referred to as generic and specific practices, respectively. Specific goals provide the basis on which fulfillment of a process area is evaluated. Both specific and generic goals are required components in the CMMI model. This means that the goal must be achieved for successful process area and maturity satisfaction. For each goal there are a number of attached practices. These describe typical solutions to or implementations of the goal which have been proven to work in other organizations. While these practices are a good starting point, they are merely expected but not strictly required. This means that if a particular organization wishes to implement different practices they are free to do so, as long as it can be shown how these alternative practices fulfill the stated goal. Finally, CMMI practices may be described in terms of sub-practices, typical work products and other notes. These are merely informative and intended to provide some initial assistance for assessing and implementing certain practices. With the exception of these informative components, CMMI model components and relations are visualized in figure 2.6.

![Figure 2.6: Staged CMMI components.](Adapted from [9])

CMMI has been found to be a useful tool for process improvement; a previous study by Gibson, Goldenson, and Kost have shown that CMMI-based process improvements can lead to significant cost savings and efficiency improvements as well as improvements to quality and customer satisfaction. In this case study of several medium and large-size companies the authors found considerable improvements to cost, planning, productivity and quality as...
a result of CMMI-based improvements efforts. While this does not show that merely using CMMI as an evaluation tool can lead to such improvements on its own, it does show that CMMI can be an effective tool in guiding process improvement efforts.

While CMMI is not a development methodology in its own right, Sutherland, Jakobsen, and Johnson [47] found that high-maturity organizations can benefit from the combined practices of Scrum and CMMI. Their study shows how agile practices can be adopted whilst maintaining CMMI compliance and that the generic goals of CMMI can be leveraged to institutionalize agile practices.

2.3.2 The SEI Framework for Software Product Line Practice

The Software Engineering Institute (SEI) proposes the Framework For Software Product Line Practice [32], which identifies practices areas and processes crucial for the successful adoption of software product lines. These practice areas are organized around three interdependent categories of essential activities: core asset development, product development, and management. Core asset development and product development are roughly analogous to the domain- and application PLE processes, respectively. Core asset development is concerned with scoping potential product lines, developing shared core assets, and planning for how products may be derived from the core assets. The product development process utilizes the work done during core asset development to produce concrete products in accordance with the product plan and to give feedback on the core assets. Finally, the management process of software product lines includes both organizational and technical aspects. The framework also highlights the importance of management at all levels being committed to the PLE effort.

All together, the SEI Framework for Software Product Line Practice defines 29 practice areas for Software Product Line Engineering (SPLE), organized around software engineering, technical management, and organizational management. Each of these categories require different (but related) skill sets and bodies of knowledge [32]. The three categories are defined as:

Software engineering “necessary for applying the appropriate technology to create and evolve both core assets and products.” ([32, p. 25])

Technical management “necessary for managing the creation and evolution of the core assets and the products.” ([32, p. 25])

Organizational management “necessary for orchestrating the entire software product line effort.” ([32, p. 25])

The practice areas corresponding to these categories are described in such detail that it is obvious which activities should be performed. Each practice area is given a general introduction and an overview of aspects particular to product lines as opposed to traditional development methods. It is also described how the practice area applies to both core asset development (domain engineering) and product development (application engineering). This is achieved both with a high-level description of how the practice area may be applied, but also in the form of example practices which may be used as a starting point for how the practice area may be applied in particular organizational context. Finally, a number of risks or pitfalls associated with the practice area are described.

However, despite the attempt to provide example practices, the practice areas are still somewhat theoretical. Simply, they lack the context of a particular situation and organization. To combat this, Clements and Northrop [7] describe a number of corresponding software product line practice patterns, which are specific patterns of problems and associated solutions that frequently show up in practice. These are essentially practical solutions to frequent problems that have been shown to work in several industrial contexts. The patterns
described by Clements and Northrop all have three components: a description of the problem, a solution related to one or more practice areas, and the organizational context in which the pattern may be applied. This body of patterns essentially allows the SEI Framework for Software Product Line Practice to cross the bridge from theory to practice.

Taken all together, the SEI Framework for Software Product Line Practice and its associated patterns thoroughly describe practices and organizational factors which are important for SPLE. However, it is distinctly geared towards describing a theoretical optimal implementation of SPLE rather than the evaluation of existing implementations. While the described practice areas can certainly be used as a point of comparison, the SEI Framework for Software Product Line Practice lacks explicit ranking or assessment methods. Thus it may be more useful for organizations just starting out with SPLE, rather than for organizations with already established product line practices.

2.3.3 Family Evaluation Framework

The Family Evaluation Framework (FEF) is a framework for assessing product line maturity, constructed especially for software product lines [27]. This framework posits that maturity may be evaluated in terms of four interconnected software development concerns:

**Business** The underlying business goals and objectives.

**Architecture** Technical aspects of how the product is designed and built.

**Process** Roles and practices that describe the development process.

**Organization** Mapping roles and responsibilities to the organizational structure.

As suggested in figure 2.7, these dimensions are all interrelated; changes to one dimension propagates to the others. However, there is also an inherent hierarchy in these concerns. This hierarchy is reflected both in the order of the acronym BAPO, and the arrows in figure 2.7. Business concerns come first and influence how the product should be built (architecture). Processes must then be established to build the application according to established goals and designs. Finally, the organization must accommodate the development process.

![Figure 2.7: BAPO model.](source)

According to van der Linden [26], the FEF incorporates aspects of several other approaches to product line evaluation: the business and architecture dimensions are inspired by the SEI Framework for Software Product Line Practice, and the process dimension is essentially an
amplification of CMMI, meaning that it uses the basic framework of CMMI but adds a number of PLE-specific goals to the more general CMMI goals. The process dimension of FEF can therefore be described as CMMI with a number of additional PLE-specific goals. For a description of CMMI, please refer to chapter 2.3.1.

An overview of the FEF is given in figure 2.8. This figure also shows that each of the four BAPO dimensions of the FEF is further subdivided into three or four more specific concerns, as well as the five maturity levels for each dimension.

![Figure 2.8: Dimensions and levels of the FEF. Source: 25.](image)

The sub-concerns of each dimension have changed somewhat from the initial version of the framework as proposed by van der Linden, Bosch, Kamsties, Känsälä, and Obbink 27. Later versions of the FEF define the following sub-concerns 25, 26:

**Business dimension**

- **Commercial** Alignment of the product line with sales and marketing objectives.
- **Financial** How PLE influences strategic financial decisions.
- **Vision** Whether PLE is part of the organization’s plan for the future.
- **Strategic** Whether PLE is an integral part of the organizations long-term business strategy.

**Architecture dimension**

- **Reuse** Extent of reuse of domain assets in application engineering.
- **Reference architecture** Whether application-specific architectures are explicitly built from available domain architectures.
- **Variability management** Explicit use of variation points and related configuration tools and processes.

**Process dimension**
2.3. Process Evaluation Frameworks

**Domain** Development of shared domain assets.

**Application** Use of domain assets to derive concrete products.

**Collaboration** Collaborative activities between domain- and application engineering.

**Organization dimension**

**Roles & responsibilities** How the organization assigns roles and responsibilities.

**Structure** Alignment of formal as well as informal organizational structure with PLE goals.

**Collaboration** Level of formal and informal collaboration across roles and business units.

Based on the specific evaluation criteria for each of these concerns, a ranking is derived for each dimension. The FEF explicitly allows for different rankings across dimensions, whilst acknowledging that the dimensions are inherently interrelated \[27\]. This ranking is done on a level from one to five, with the process dimension using the same rankings as CMMI (see chapter \[2.3.1\]) with a few PLE-oriented amplifications. These amplifications are described in appendix \[A\].

### 2.3.4 SCAMPI

The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is a formal approach to CMMI appraisal and evaluation, intended mainly for SEI-certified appraisers \[48\]. However, the methodology and general approach may also be of use to other stakeholders involved in the appraisal process. SCAMPI covers nearly all parts of the appraisal: of particular interest for this thesis are the core activities for conducting a CMMI appraisal. This is organized around three phases, each with a number of sub-processes \[48\]:

**Plan and prepare** Understanding the context and business needs of the organization. Information is collected to match appraisal objectives with business objectives. A strategy for how to collect data is formed.

- Analyze requirements
- Develop appraisal plan
- Select and prepare team
- Obtain and inventory initial objective evidence
- Prepare for appraisal conduct

**Conduct appraisal** Collecting data and generating results. Evaluation ratings are generated from gathered data.

- Prepare participants
- Examine objective evidence
- Document objective evidence
- Verify objective evidence
- Validate preliminary findings
- Generate appraisal results

**Report results** Provide and present credible results.

- Deliver appraisal results
- Package and archive appraisal assets

For each of these processes, several sub-activities are defined and described along with entry criteria and input- and output artifacts.
2.4 Methodology

This chapter describes a number of theories on how a qualitative case study may be performed. This serves as an overview of applicable theories and approaches, and on the advantages and disadvantages of these. The method according to which the study was performed is detailed in chapter 4.

2.4.1 Planning and conducting a case study

A compelling approach to designing a case study is given by Runeson and Höst due to their explicit guidelines for research in the field of software engineering. Their approach to software engineering research is centered around five core activities:

1. Designing the case study
2. Preparing for data collection
3. Collecting evidence
4. Analyzing collected data
5. Reporting conclusions

Runeson and Höst give practical advice and guidelines for each of these activities. They also highlight the flexible nature of case studies when compared to other types of empirical approaches; a case study should be performed incrementally, with data collection, analysis methods, and scope being adjusted during the course of the study if necessary. Before describing these steps in further detail, let us briefly consider alternative research approaches to a typical case study. Runeson and Höst describe four different approaches to empirical research:

- Survey
- Experiment
- Action research
- Case study

According to Runeson and Höst, a survey takes a quantitative approach to data collection. The gathered data is standardized and the overall study design is fixed, meaning that the research focus and method is not subject to change during the course of the study. They contrast this with an experiment which is more explanatory in nature and attempts to explain the impact that one studied variable has on another. A proper experiment isolates variables as much as possible and assigns subjects to different groups (e.g. control group or experiment group) at random. They go on to describe that action research sees the researcher take a more involved role in the organization or group that is being studied, with the researcher purposefully influencing or changing some part of the system being studied. The researcher is involved in the change process, whilst simultaneously describing and evaluating the process through qualitative means. Finally, they describe how by replacing the active researcher in the action research approach with a more exploratory role we finally land in the realm of a case study. The role of the researcher in a case study is to explore what is happening and seek new insights through qualitative means. Runeson and Höst recommend the use of case studies in software engineering research because such research is often multi-disciplinary across several fields where case studies already are a proven practice. This includes the social aspect of software development, and how software development is carried out by individuals as well as groups.

The remainder of this chapter will give brief descriptions of the activities and considerations that should be made for the five steps of a case study as defined by Runeson and Höst.
2.4. Methodology

Case study design

Robson [37] describes five components of a research plan that should be kept in mind when designing and carrying out a research project:

**Purpose** What is the reason for this study?

**Theory** What are the theoretical underpinnings of the study?

**Research questions** What are the specific questions that the study is trying to answer?

**Methods** How will the research questions be answered?

**Sampling strategy** How and from whom will the data be collected?

Aside from such a research plan, Runeson and Höst [38] also highlight the need for a detailed *case study protocol*: a living document that details the current plan for the case study. Ethical consideration should also be made in the design phase [38].

**Preparation for data collection**

Before beginning to collect research data, procedures and protocols for how the data collection will be performed should be defined [38]. This includes procedures for updating the case study protocol, how multiple data sources will be triangulated and an evaluation of proposed methods and measurements [38].

**Collecting evidence**

Data that can be corroborated by other sources of information is more reliable than an interpretation of a single data source [38]. Corroborating data may be in the form of different viewpoints and subjects, but also in the form of cross-referencing data collected through different means. Lethbridge, Sim, and Singer [24] define three levels of data collection methods:

**First degree** Direct involvement, e.g. interviews, questionnaires or participating in the team.

**Second degree** Indirect involvement, such as by monitoring or recording subjects.

**Third degree** No involvement with subjects. Study is only concerned with product artifacts or documentation.

Runeson and Höst [38] describe these as generally descending in terms of effort for the researcher: first degree methods require a great deal of involvement and effort on part of the researcher, while second and third degree methods are considerably less taxing. However, they also note that the usefulness of the collected data is also greater with first degree methods than with less involved methods. Interviews are a fairly typical approach to data collection in case studies [38]: this is covered in greater detail in chapter 2.4.2.

**Analysis of collected data**

Runeson and Höst [38] base their approach to qualitative data analysis on the four different approaches to qualitative analysis defined by Robson [37]:

**Quasi-statistical** A formalized approach to data analysis, typically built on statistical analysis of word or phrase frequencies.

**Template** Data is coded and organized based on key codes derived from the research questions.
2.4. Methodology

**Editing** More flexible than template approaches with codes being based on the researcher’s interpretation of the data.

**Immersion** Least formal approach. Data is analyzed purely through the researcher’s own interpretation and intuition.

As previously described, Runeson and Höst \(^38\) recommend the use of template or editing approaches to data collection in software engineering research. They discard immersion techniques for their inherently unscientific nature and argue that quasi-statistical approaches are typically difficult to implement when working with software engineering documents and interviews.

**Reporting conclusions**

Finally, Runeson and Höst \(^38\) define a few guidelines for reporting the findings of a case study. They highlight the importance of balancing the clarity of the report with the subjects’ right to integrity. This balancing act can also be extended to other aspects of the report. The validity of the study depends on how the study was executed, but it is typically not appropriate to include every single execution detail. Similarly, the analysis and conclusions must be supported by snapshots of the raw data \(^38\).

2.4.2 Interview techniques

As previously stated, interviews are a common qualitative data gathering technique in case study research \(^38\), \(^12\). Robson \(^37\) defines three basic types of interviews: **fully structured interviews**, **semi-structured interviews**, and **unstructured interviews**. A fully structured interview is designed around a fixed set of mainly open-ended questions \(^37\). According to Runeson and Höst \(^38\), this approach is mainly used to find explicit links between concepts, but lacks the exploratory aspects of less rigid methods. A more exploratory approach is instead offered by semi-structured and unstructured interviews. These interview techniques allow the researcher to change topics and questions on the fly, based on what is found during the interview \(^37\). The main difference between these two techniques is in the use of an interview guide: semi-structured interviews use an underlying interview guide with a list of questions (although questions may be changed, removed or added at the researcher’s discretion), while an unstructured interview is merely centered around a general topic \(^37\).

2.4.3 Trustworthiness

In the realm of quantitative research, the concepts of reliability and validity are typically used when discussing research quality. However, these concepts cannot be applied in the same way to qualitative research, such as case studies \(^17\). Instead, the concept of **trustworthiness** is typically used when referring to qualitative studies \(^46\). Guba \(^18\) defines four aspects of trustworthiness in qualitative research:

- Credibility
- Transferability
- Dependability
- Confirmability

These are expanded upon by Shenton \(^16\) who suggests a number of provisions for each aspect that should be made to ensure that the criteria for trustworthiness are met. The remainder of this chapter will describe these aspects in further detail and give an overview of the methods proposed by Shenton.
2.4. Methodology

**Credibility**  Credibility in qualitative research primarily revolves around the internal validity of the study, that is whether the findings correspond to reality and a relationship between cause and effect [46]. Shenton suggest using methods such as prolonged engagement with the studied group, random sampling of subjects, triangulation using different types of data and ethical data collection to ensure credibility in qualitative research.

**Transferability**  In qualitative research, transferability corresponds to the external validity of the study. This is whether the findings can be generalized to other groups and contexts. According to Shenton, qualitative research is often impossible to generalize across different situations and populations. Guba [18] acknowledges this limitation and states the study should be explicit about contextual factors and describe these in detail. Shenton suggests that the following factors should be described by the researcher:

- Number of studied organizations and where they are based
- Restrictions in types of participants
- Number of participants
- Data collection methods
- Number and length of data collection sessions
- Time period of data collection

**Dependability**  Since qualitative research such as a case study is necessarily a snapshot of an organization at a particular point in time, Shenton argues that such studies are typically not repeatable in the classical sense. Instead, dependability in qualitative research revolves around describing the research process such that the study can be replicated in terms of execution but not necessarily results. For this the researcher should detail the planning and execution of the study, data collection methods, and reflections on the effectiveness of the research process.

**Confirmability**  The confirmability of qualitative research is built on the separation of the researchers own bias from the findings. Shenton describes how triangulation and detailed reasoning and reflection about the theoretical underpinning of the study is important to ensure confirmability. This includes discussing and motivating the chosen method and being open about alternative approaches. A clear audit trail can also be useful in ensuring confirmability by detailing how the study was executed and conclusion drawn in a step-by-step manner.
Saab develops and maintains three main types of simulator environments in the realm of aircraft development and training: development simulators, training simulators, and simulators for marketing and prototyping. Previous studies and trials have shown the value of introducing Product Line Engineering (PLE) processes for these products and Saab is currently working towards moving more products over to product lines as well as increasing the maturity of existing PLE implementations. This move towards PLE can be seen as an expansion of the existing “design-once” philosophy. They state that:

"Saab delivers advanced operational support systems and training system media made to reflect the weapon system’s current configuration. Saab has established a development process where all requirements for the entire weapon system are captured early, thus influencing its design right from the start. The ‘design once’ approach, common to all tools and software used in developing the real aircraft, ensures that any changes to the aircraft are automatically reflected in the support and training systems.” (39)

This proto-PLE philosophy can be seen as the main guiding principle for reuse at the outset of this thesis. One of the main challenges to overcome in the introduction and expansion of PLE in this context is that these projects and components differ in purpose and requirements, and are often aimed at different stages of the aircraft development life cycle. This means that there is often great variability in how work is conducted in the various project groups and teams responsible for different simulator products and components. Consequently there is no one-size-fits-all solution to how new processes and methods may be introduced to the various simulator projects. As such, there exists a need for formal evaluation of the different development and PLE processes used across all of these product lines.

As previously described there are three types of simulators that are included in this study. However, the development simulators are further divided according to aircraft generation. This gives us a total of four top-level product lines to investigate: Development simulators C/D and E/F (corresponding to the Saab 39 Gripen models C/D and E/F, respectively), training simulators, and marketing and prototyping simulators. These products are not at the time of writing developed according to established PLE processes, but a formal PLE maturity evaluation is being conducted in order to guide future PLE efforts. The introduction and feasibility of PLE at Saab has been covered by Andersson, Herzog, and Ölvander [1].

24
Andersson, Herzog, and Ölvander describe how modeling and simulations-based development methods may be combined with PLE practices to maximize reuse and minimize development effort. Their proposed methodology was evaluated in an industrial context by applying it to the simulators maintained at Saab. Their work present how PLE practices may be implemented via the use of custom tools and metadata stored together with source code. They describe how they developed a tool that can be used to select and configure components used in a particular simulator configuration. This tool also supports custom rules and constraints. Andersson, Herzog, and Ölvander show that the proposed methodology was viewed positively by practitioners and that it led to considerable time savings, especially in the release process. They also describe how practitioners highlighted the importance of improvements to product quality over time-saving measures. Andersson, Herzog, and Ölvander clearly show that certain PLE practices can have a positive effect on large-scale simulator development by promoting reuse and enabling late binding of simulator models.

In this particular example, the methodology proposed by Andersson, Herzog, and Ölvander shows that the introduction of PLE practices and tools resulted in concrete improvements to the simulator configuration process and simplified the creation of release artifacts. The tool described by Andersson, Herzog, and Ölvander essentially fills the role of the PLE configurator as illustrated in figure 2.2. By enabling automated or tool-assisted feature selection, such a configurator is a key component in a practical PLE implementation. The success of this approach shows that PLE practices do indeed have some practical potential to save time and increase product quality when applied to the development of large-scale simulators. While this methodology may not describe a fully-fledged PLE-driven development process, it does show that PLE practices can be successfully applied to specific problem and processes at particular points in the software development life cycle.
As previously discussed, this thesis is based on a case study conducted on the development processes used to construct large-scale simulators at Saab. This chapter describes the overall design of the performed case study, the interview methodology and interview guides, and how the process evaluation frameworks described in chapter 2.3 were used to evaluate the gathered data. The underlying theory (and relevant alternatives) that motivated the chosen approach is discussed in chapter 2.4.

4.1 Designing the case study

The case study was designed with consideration to the approach suggested by Runeson and Höst [38]. A continuously updated case study protocol was used to track the goal and design of the case study. This protocol and the initial case study plan were constructed according to the recommendations for software engineering research given by Runeson and Höst [38]. The case study plan consisted of six distinct elements:

Objective To evaluate the effectiveness of PLE processes to construct large-scale simulator products. This was evaluated in terms of process maturity and goal fulfillment as well as satisfaction with PLE processes.

The case Three different product lines were investigated. These corresponded to two major simulator tracks at Saab: development simulators and simulators for marketing and prototyping. For each product line a process maturity rating was derived according to the FEF method. In addition to this, subjective opinions on and experiences with PLE processes were gathered.

Theory PLE was theorized to be suitable for construction of simulator products due to alignment of theoretical benefits (e.g. strong potential for software reuse) and previous studies showing PLE to be supported in architecture as well as tools. The theoretical framework for process evaluation was based on the FEF and CMMI.

Research question The study was focused on a single research question: To what extent are PLE development practices suitable for developing large-scale simulators?
Methods The main data gathering method was a series of semi-structured interviews. A number of interviews were conducted for each product line, with each interview subject having a different role in the organization. These roles included line managers, technical leaders and engineers.

Selection strategy Interview subjects were selected according to availability and on recommendation from their respective line manager. A total of six subjects were chosen with various roles: engineers (line A), technical leaders (lines A and B), and line managers (lines A, B and C).

4.2 Conducting interviews

Interviews were conducted in a semi-structured fashion, as per the recommendation from Runeson and Höst [38]. The same interview guide was used for all interviews, to ensure a basic level of topical cohesion between interviews. The interview guide can be found in appendix B. While semi-structured interviews does allow for a certain variability in interview conduction, the following basic format was used for all interviews:

1. Introducing the study and the objectives.
2. Describing how the collected data will be used.
3. Describing the steps taken to anonymize collected data.
4. Asking for consent to record interview for later transcription.
5. Asking introductory questions about background, role, etc. according to the interview guide.
6. Asking the main questions according to interview guide, allowing for some variation and exploration other topics that are brought up during the interview.
7. Summarizing the major findings and allowing the subject to make clarifying remarks or corrections.

This format was inspired by the interview phases described by Runeson and Höst [38]. The main questions pertaining to the CMMI evaluation were organized around the CMMI process areas rather than by level. This was intended to keep the discussion focused around one particular topic at a time, rather than coming back to the same topic multiple times. The same goes for the three process dimensions: domain engineering, application engineering, and collaboration. For each process area a number of questions were formulated to evaluate its maturity level across all three dimensions. Once a particular process area had been exhausted, the interview proceeded with the next process area. This basic structure is illustrated in figure 4.1.

In terms of specific versus general questions, the interviews followed the hourglass approach described by Runeson and Höst [38]. This means that the interview started out with quite open questions about each process area, followed by more specific questions about goals and practices for each process area and FEF dimension. Finally, the interview opened up once again with questions about the subject’s impression of the product line and the associated processes as a whole.

A total of six interview sessions were conducted, ranging from 1.5 to 2.5 hours in length. These were conducted during a time span of two months to accommodate the scheduling needs of the subjects.
4.3 PLE process evaluation

Evaluation of PLE processes was conducted according to the FEF. This decision was grounded in the holistic nature of the FEF and available detailed documentation of CMMI. It was also based on the success of previous case studies using FEF. Chapter 2.3 provides an overview of other frameworks that may be used to evaluate process maturity: The SEI Framework for Software Product Line Practice, and SCAMPI. The SEI Framework for Software Product Line Practice was rejected because it lacks explicit ranking or assessment methods, meaning that it would have been more difficult to make comparisons across product lines. The SCAMPI framework does not provide a maturity evaluation per se, but rather sets guidelines for how the evaluational may be performed. This was not explicitly followed because it is aimed at professional appraisers rather than academic study. As such, the approach described in this chapter was used instead.

FEF builds on the CMMI framework with a number of amplifications as described in appendix A. This method gives an overall ranking of process maturity, based on the maturity levels of 22 different process areas along three dimensions: domain engineering, application engineering, and collaboration between domain and application engineering. A more detailed description of the FEF method is found in chapter 2.3.3.

The evaluation was performed through the use of in-depth interviews with key product line personnel as described in chapter 4.2. To analyze the gathered data, the editing technique described in chapter 2.4.1 was used. This approach means that the interviews were transcribed and then coded according to the themes and sentiments that came up during the course of the interview. Codes were not predetermined, but rather created with the researcher’s own interpretation of the data in mind. This was done on a detailed level, with passages being allowed to be tagged with multiple codes. Care was taken to tag passages based on their actual content rather than the current question or process area. This allowed for the topic to drift over time with the coded data reflecting the changing topic. The process of coding and analyzing interview data is illustrated in figure 4.2. The codes were used to tag interesting quotes and discussions in the transcripts, which allowed similar quotes to be grouped together.

The coded data was then used to perform the FEF evaluation. Similar quotes were grouped according to process area and general sentiment. This data was then used to construct a picture of existing processes and issues. The maturity evaluation was done by comparing the processes
as they were described in the interviews with the goals and practices required by CMMI together with the FEF amplifications described in appendix A. The evaluation was mainly based on the CMMI process area goals rather than specific practices. This reflects the fact that only the goals themselves being required, with specific practices merely being expected. The evaluation takes this into account by focusing on goals instead of practices. A rating of Fulfilled, Partly fulfilled, or Not fulfilled was assigned separately to the specific goals, general goals and FEF amplifications of each process area for each product line. A rating of Fulfilled means that the product line was deemed to fulfill the specified goals of the process area. Partly fulfilled means that some but not all aspects of the goals were fulfilled. Not fulfilled means that no processes existed for that process area, or that existing processes were not found to fulfill any aspects of the CMMI goals.
This chapter presents the data that was gathered during the interviews and the resulting FEF maturity ratings. The methodology is described in chapter 4 and will not be covered in further detail here. Since the interviews were conducted in Swedish, any quotes from interview subjects referenced in this chapter will be translated into English. Individual results will be presented for each product line, three in total. These product lines will be referred to as product line A, B and C.

5.1 Codes

As described in chapter 4.3, an editing approach was used to derive codes from the interview transcripts. These codes were then used to group relevant quotes and analyze the data. The codes extracted from the interview data can be found in table 5.1. Note that the codes are not rigorously described in this chapter, since they were relevant mostly for the underlying analysis of the interview data.

5.2 FEF Maturity Ratings

This chapter presents the results of the FEF process maturity ratings for each investigated product line. An overview of the results may be seen in table 5.2. This table presents a detailed breakdown of which CMMI goals and FEF amplifications were fulfilled for each product line. CMMI goal fulfillment is presented separately for specific and generic goals. The distinction between these is described in chapter 2.3.1. Only if all three factors (i.e. specific goals, generic goals and FEF amplifications) are fulfilled should the process area as a whole be considered fulfilled.

Product Line A

The overall FEF maturity rating of this product line was determined to be 1: Initial, with partial fulfillment of the specific goals for maturity rating 2: Managed. Of the specific goals of the process area corresponding to this level, requirements management and process and product quality assurance were found to be partially fulfilled. The remaining process areas were all fulfilled in terms of their specific goals.
### Table 5.1: Extracted codes.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>Subject’s background and work experience at Saab</td>
</tr>
<tr>
<td>CAR</td>
<td>Causal Analysis and Resolution</td>
</tr>
<tr>
<td>CM</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>Communication</td>
<td>Communication within and across teams</td>
</tr>
<tr>
<td>Cooperation</td>
<td>Cross-team cooperation efforts</td>
</tr>
<tr>
<td>Culture</td>
<td>Organizational culture</td>
</tr>
<tr>
<td>DAR</td>
<td>Decision Analysis and Resolution</td>
</tr>
<tr>
<td>Dependencies</td>
<td>External dependencies</td>
</tr>
<tr>
<td>Documentation</td>
<td>Documentation processes</td>
</tr>
<tr>
<td>IPM</td>
<td>Integrated Project Management</td>
</tr>
<tr>
<td>Legacy</td>
<td>Legacy systems and processes</td>
</tr>
<tr>
<td>MA</td>
<td>Measurement and Analysis</td>
</tr>
<tr>
<td>Management</td>
<td>Opinions and experiences of upper management</td>
</tr>
<tr>
<td>OPD</td>
<td>Organizational Process Definition</td>
</tr>
<tr>
<td>OPF</td>
<td>Organizational Process Focus</td>
</tr>
<tr>
<td>OPP</td>
<td>Organizational Process Performance</td>
</tr>
<tr>
<td>OT</td>
<td>Organizational Training</td>
</tr>
<tr>
<td>Planning</td>
<td>Planning efforts</td>
</tr>
<tr>
<td>PLE Experience</td>
<td>Subject’s experience of PLE</td>
</tr>
<tr>
<td>PLE Opinion</td>
<td>Subject’s opinion of PLE</td>
</tr>
<tr>
<td>PM</td>
<td>Project Monitoring</td>
</tr>
<tr>
<td>PP</td>
<td>Project Planning</td>
</tr>
<tr>
<td>PPQA</td>
<td>Process and Product Quality Assurance</td>
</tr>
<tr>
<td>Product</td>
<td>Product descriptions</td>
</tr>
<tr>
<td>PU</td>
<td>Product Integration</td>
</tr>
<tr>
<td>QPM</td>
<td>Quantitative Project Management</td>
</tr>
<tr>
<td>RD</td>
<td>Requirements Development</td>
</tr>
<tr>
<td>Reflection</td>
<td>Systematic reflection efforts</td>
</tr>
<tr>
<td>REQM</td>
<td>Requirements Management</td>
</tr>
<tr>
<td>Responsibility</td>
<td>Defined areas of responsibility</td>
</tr>
<tr>
<td>Reuse</td>
<td>Sharing and reuse across products/components</td>
</tr>
<tr>
<td>RSKM</td>
<td>Risk Management</td>
</tr>
<tr>
<td>SAM</td>
<td>Supplier Agreement Management</td>
</tr>
<tr>
<td>Support</td>
<td>Support resources</td>
</tr>
<tr>
<td>Tools</td>
<td>Tooling support</td>
</tr>
<tr>
<td>TS</td>
<td>Technical Solution</td>
</tr>
<tr>
<td>VAL</td>
<td>Validation</td>
</tr>
<tr>
<td>VER</td>
<td>Verification</td>
</tr>
</tbody>
</table>
The specific goal of the requirements management process areas was considered partially fulfilled. The evaluation was based on the finding that while requirements are gathered and managed, they are not documented in a formal fashion. According to one respondent, “The simulators that we have are old and no one knows where the requirements that existed back then come from. Everything has just worked and that’s that. Those who developed [the requirements] back then are not with the company any more.” As such, the process for requirements management was not deemed to be conducted in a documented fashion.

When evaluating the process and product quality assurance process area it was found to be in partial fulfillment of both its specific goals (SG1: "Objectively evaluate processes and work products" and SG2: "Provide objective insight"). While it was found that process quality was indeed discussed internally, it was not evaluated regularly or based on objective measurements. According to one respondent, they “have had a retrospective activity” but that it was “not performed regularly.” The same respondent affirmed that while the team does communicate process issues, they are not evaluated systematically.

Moving on to the process areas required for the CMMI maturity rating of 3: Defined, requirements development, technical solution, product integration, organizational training, integrated project management, risk management and decision analysis and resolution were all found to fulfill their stated specific goals. The verification, validation and organizational process definition process areas partly fulfilled their specific goals while only the organizational process focus and decision analysis and resolution processes did not fulfill any specific goals.

The specific goals of the verification and validation process areas were both considered partly fulfilled. The verification process area was found to lack defined process for performing peer reviews (SG2: "Perform Peer Reviews"). According to one respondent, peer reviews are generally performed on most software components but the extent of those reviews were described as changing quite drastically based on who performs the review and who writes the code:

"Sometimes [we perform code reviews]. Not always. It depends on the person doing it I would say. We have several [co-workers] who are considered very talented and have a lot of experience, and those don’t get reviewed as often as those who are newer."

The validation process was found lacking in planning (SG1: "Prepare for validation") and only in partial fulfillment of SG2: "Validate Product or Product Components”. According to the respondents, the validation process is not performed as a defined activity per se, but aspects of validation was found in the verification testing activity. The respondents described how the customer is invited to test the product and give a final verdict on how it performs in its intended environment. One of the respondents also described a fairly lengthy period where components are trialled in unofficial simulator builds. Users are encouraged to use these unofficial builds in order to evaluate the state of the simulator components before formal system testing is performed. While this was not explicitly referred to as validation testing, this process can nevertheless be considered to fulfill the basic purpose of ensuring that the product will perform its intended function in its intended environment. Thus, the validation process area was considered to partially fulfill its specific goals.

The organizational process definition process area was found to be partly fulfilled. This evaluation was based on the finding that while there is a standardized set of processes in terms of the organizational management system, these process descriptions are largely disconnected from the processes implemented internally. According to one respondent:

"I don’t really recognize [our processes] from the management system or anything like that. Maybe the project management process and things like that."

Another respondent stated:
The decision analysis and resolution process was found unfulfilled due to no such methods or practices being used internally.

The remaining four process areas corresponding to maturity levels four and five were all found not to be fulfilled.

Common to all process areas was the lack of complete fulfillment of the CMMI general goals. The partly fulfilled goals generally fell short of general practices 2.8 ("Monitor and control the process") and 2.9 ("Objectively evaluate adherence") or a general lack of documentation describing and planning the process.

The FEF amplifications were found to generally be unfulfilled. However, the amplifications for the product integration process area was found to be fulfilled. The amplifications for the project planning and integrated project management process areas were found to be partially fulfilled.

The evaluation of partial fulfillment of the project planning amplifications was based on the finding that while domain assets are considered for reuse in the planning stage, the remaining amplifications such as defining variability and analyzing the risk of dependency on domain engineering were not described as begin performed in a defined manner.

The product integration process area amplifications were found to be fulfilled. This requires a roadmap of future products and enhancements to be maintained and for plans to be made in advance detailing how products are to be integrated. According to one respondent, they “receive specifications or documents describing how the integration should be done” and integrations are generally preceded by a planning process.

The integrated project management amplifications were found to be partially fulfilled. These amplifications require planned and existing assets to be communicated and for applications assets to be identified as potential domain assets. According to one respondent, these plans are communicated twice weekly in regularly scheduled meetings where development activities are synchronized across teams. However, it could not be established that application assets are identified as potential domain assets in a defined fashion. Thus, integrated project management amplifications were evaluated as partially fulfilled.

Remaining process area amplifications were not found to be fulfilled, neither completely nor partly.

**Product Line B**

The overall FEF maturity rating of this product line was determined to be 1: Initial. Of the seven process areas required for a rating of 2: Managed, requirements management, project planning, project monitoring and control, supplier agreement management, and configuration management were all found to fulfill the specific CMMI goals for these process areas. The process and product quality assurance process area was found to only partly fulfill both its specific goals (SG1: "Objectively evaluate processes and work products" and SG2: "Provide objective insight"). Firstly, the respondents stated that while certain objective evaluations of work products are performed in the verification phase, no such objective evaluations are performed for processes. When asked about this, one respondent succinctly stated that “we don’t really check that.” Secondly, not enough objective data about both processes and work products is gathered to be able to properly provide objective insight. This lack of objective data is mirrored in the lack of fulfillment of the specific goals of the measurement and analysis process area. According to one respondent they perform “no type of data collection”. As
such, the specific goals of the measurement and analysis process area could not be considered fulfilled.

For the process areas corresponding to maturity level 3: Defined, only decision analysis and resolution was complete unfulfilled in terms of specific goals. According to all respondent, no decision analysis methods are applied in any capacity. The validation process area was determined to not fulfill SG1: "Prepare for Validation" and only partly fulfill SG2: "Validate Product or Product Components". This evaluation was based on the finding that while there was no formal validation activity performed for this process area, certain aspects of the verification process could conceivably be considered to belong to the validation process area. This was pointed out by one of the respondents who stated:

"What we’re doing is much closer to validation than verification. We sort of roughly say that [a certain component] does what it’s supposed to."

The process was also described as being geared towards letting users test the product(s) and thus evaluate their usefulness. According to the respondents, this is performed as part of the verification testing phase. However, it also fills the role of simple validation testing since it captures the basic aspects of ensuring that the product performs its intended function in its intended environment. Thus, even if this is not seen as a formal kind of validation testing by the organization, it does partly fulfill SG2 ("Validate Product or Product Components") of the validation process area. However, SG1 ("Prepare for validation") could not be considered fulfilled since no deliberate validation activity was determined to take place.

The organizational process focus process area falls short of fulfilling all specific goals due to only partly fulfilling SG1: "Determine process-improvement opportunities". The evaluation of partial fulfillment was based on the finding that even though process improvement sometimes takes place, determining opportunities for such improvements was not found to be done in a managed fashion. One respondent described the identification process as:

"There is some competition between process improvement and the day-to-day operation because the need for process improvement usually comes in a kind of panic. It’s usually very sudden. The economic prerequisites aren’t really there to have a philosophy of moving in a certain direction. Nearly all improvements are rather done because of some urgent need."

The organizational process definition was found to be partly fulfilled, due to only partly fulfilling its single specific goal (SG1: "Establish Organizational Process Assets"). The evaluation of partial fulfillment was based on the finding that while certain processes are defined by the operations management system, the general consensus among the respondents was that these process definitions had little to do with the department-specific processes and activities. However, the perception of the common process assets also differed somewhat between different respondents. One respondent stated that:

"There is the operations management system, so there is a rulebook of sorts. [...] I think that’s translated fairly well, but I haven’t been involved enough to say. But yes, you usually refer to that."

However, another respondent described that processes are usually defined internally:

"I would say we have come up with [the processes] ourselves. I think our organization is a bit special in this case. [The department] is to start with not a company of course, and everything is under development. Nothing is really standardized, but can be changed and so on. And that also applies to certain processes. That in combination with certain processes not being used so often has resulted in us having invented them ourselves so to speak."
This view was shared by a third respondent:

"I don’t really know how things look externally, but from what I’ve seen internally
I have a hard time believing that [our processes] come from anywhere else."

Finally, the risk management process area was considered to be partly fulfilled. This
evaluation was based on a lack of fulfillment of SG1 ("Prepare for Risk Management") and only
partial fulfillment of SG3 ("Mitigate risks"). While the respondents all described how some risk
management processes and activities were in place, they lacked a consistent risk management
strategy. For instance, respondents did not describe the risk management plan as having
defined risk sources and categories or a consistent risk management strategy. Moreover, while
risks were described as being identified, respondents described the strategies for mitigating
risks as ad-hoc:

"When things have to be done they have to be done. There aren’t many strategies
for dealing with [risks], it’s very ad-hoc. It’s not like you can move over to plan B."

The remaining four process areas corresponding to maturity levels four and five were all
found not to be fulfilled.

Common to all process areas was the lack of complete fulfillment of the CMMI general goals.
The partly fulfilled goals generally fell short of general practices 2.8 ("Monitor and control
the process") and 2.9 ("Objectively evaluate adherence") or a general lack of documentation
describing and planning the process.

FEF amplifications were generally absent for the majority of process areas. The amplifica-
tions that were partly fulfilled were the amplifications to project planning, configuration man-
agement, requirements development, technical solution, product integration and integrated
project management. The FEF amplifications to project planning were considered partially
fulfilled due to a mindful approach to reuse in the project planning phase. However, it could not
be considered completely fulfilled due to a lack of separation between domain and application
engineering activities.

The configuration management amplifications were also considered partially fulfilled. Ac-
cording to one respondent, there are processes in place for product-specific changes to lead to
changes to the underlying reusable assets. However, the remaining configuration management
amplifications were not found to be fulfilled.

The requirements development process area amplifications were considered partially ful-
filled due to the product line considering a single customer in the application engineering
phase. According to one respondent the requirements are generally geared towards a single
simulator rather than on the product line as a whole, but the capabilities of the product line as a
whole is used when communication with the customer. However, the remaining requirements
development amplifications were not found to be fulfilled.

When evaluating the FEF amplifications to the technical solution process area, these were
found to be partly fulfilled. This evaluation was based on the finding that while assets are
reused during the implementation phase it was not found that variability is included as an
operational concept for the domain. Indeed, there is little separation between domain and
application engineering.

The product integration amplification were the only FEF amplifications found to be ful-
filled for this product line. These amplifications are concerned with maintaining a “roadmap
of future products and product enhancements” ([25], p. 92), planning and timing product
transfers, and supporting the “integration of domain and application assets” ([25], p. 92).
Although no meaningful separation between domain and application engineering was found,
these activities were found to be performed in the project planning and monitoring processes.
One respondent described the basic integration planning process thus:
"You have [a meeting] where you plan for these function to be integrated. They might have to be developed or be acquired from a third party. And when that’s decided you plan when and how it should be done. You might have to test it beforehand, perhaps in a software-based simulator or an experimental build in the hardware-based simulator. Then you have a trial period where you integrate the new software and perform integration tests, checking that everything seems to work. The next step is a systems test where you test the software together with everything else and make sure that nothing is broken. Then you release the simulator and inform the users that this new function now is available."

The final process area with partly fulfilled amplifications is the integrated project management process area. This process area received the evaluation of partly fulfilled due to not properly communicating existing assets. According to one respondent, it is not properly communicated which assets are available for reuse:

"I think [we reuse components] in the simple cases when you see an obvious opportunity for reuse. But that analysis is perhaps not thorough enough or is not supported well enough. [A reusable component] might be available somewhere, but you don’t know where to look or even know that it exists."

According to this description, available domain assets could not be considered to be not properly communicated. However, it was found that there exists processes for changes to application assets to propagate to domain assets. Thus, the integrated project management amplifications were found to be partly fulfilled.

Remaining process area amplifications were not found to be fulfilled, neither completely nor partly.

**Product Line C**

The overall FEF maturity rating of this product line was determined to be 1: *Initial*. Two of the process areas required for a rating of 2: *Managed* were found to be fulfilled in terms of their specific goals: project planning and configuration management. Two more process areas were found to have partial fulfillment of their specific goals: supplier agreement management and process and product quality assurance. The remaining process areas were not found to fulfill their specific goals.

The supplier agreement management process area was found to fulfill SG1: "Establish Supplier Agreements" by having reportedly defined dependencies for external dependencies. The definition of ‘supplier’ was relaxed slightly since these dependencies were described as being in-house and thus not following a traditional client-supplier relationship model. As such, specific CMMI practices such as review of commercial off-the-shelf components were deemed to not be applicable. Thus, supplier agreements were described as being more casually defined than with external dependencies. However, the second specific goal (SG2: "Satisfy Supplier Agreements") was found to only be partially fulfilled. This was based on the finding that the product line “doesn’t have a formal delivery activity”, as described by one of the respondents. This respondent also described several areas of improvement for the supplier agreement management process area:

"I would like to see a more formal delivery process. [...] We need to have a clear idea of what’s been delivered, what the status of the delivery is, what the declaration is based on, and what the information classification is."

The process and product quality assurance process area was also found to be partially fulfilled. This evaluation was based on the finding that while some process and product quality evaluations are performed, these are mostly not based on objective measures and is not a well-defined process:
"That process isn’t defined. We have a number of manual tests that can be adapted to new developments fairly easily. But that just manual, there are no automatic tests to ensure quality. Then you sit down with the customer and look at it together."

When asked about process quality, they said that their approach to process quality evaluation is “not deliberate. There is nothing defined about it. What we have is that sometimes, maybe not continuously but sometimes, we review our working processes.” Based on these statements, the specific goals of the process and product quality assurance process areas were deemed to be partly fulfilled.

Of the process areas required for the next maturity level (3: Defined), requirements development, verification, organizational process focus and organizational training were found to fulfill their respective specific goals. The technical solution, product integration and integrated project management process areas were considered partially fulfilled. The remaining process areas of validation, organizational process definition, risk management and decision analysis and resolution were found to be unfulfilled.

The technical solution process area was considered partially fulfilled due to a lack of documented development practices. One respondent described how the design process is mainly left to individual developers without proper documentation or support. According to this respondent, there is also no systematic approach to reuse. The documentation that does exist on the development process “describes how you proceed from starting development, checking in code and how it’s reviewed and approved. But it’s very much focused on how code is checked in and version controlled. So, really, it starts in the middle of the development process.”

The product integration process area was found to be partially fulfilled due to not fulfilling specific goals SG1 (“Prepare for Product Integration”) and SG2 (“Ensure Interface Compatibility”). When describing the planning process, one respondent stated that:

“There is a lack of documentation. It’s not a very clear process. If there is a need to integrated we usually just talk it through and see if it’s possible, and then we do it.”

The same respondent described how interfaces are not reviewed systematically:

“I don’t think there is [a process for reviewing product interfaces]. oftentimes I think it’s just up to the people who have been working here a long time. And if it’s something external then you have to take a look at it, but I wouldn’t say that we have any tooling support or documentation about it. You have to go more with your gut.”

Finally, the specific goals of the integrated project management process areas were found to be partially fulfilled. Because of the lack of defined processes, SG1 (“Use the Project’s Defined Process”) could not be considered fulfilled:

“I think [our processes] are completely freestyle. […] It’s been allowed for a long time to work in this way. There haven’t been any requirements that you should work according to the defined processes.”

The remaining four process areas corresponding to maturity levels four and five were all found not to be fulfilled.

Common to all process areas was the lack of complete fulfillment of the CMMI general goals. The partly fulfilled goals generally fell short of general practices 2.8 ("Monitor and control the process") and 2.9 ("Objectively evaluate adherence") or a general lack of documentation describing and planning the process.

FEF amplifications were found not to be fulfilled for all process areas.
### 5.2. FEF Maturity Ratings

Table 5.2: Product line maturity ratings.

<table>
<thead>
<tr>
<th>Process Area</th>
<th>CMMI maturity level</th>
<th>Product Line</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SG</td>
<td>GG</td>
</tr>
<tr>
<td>Requirements Management</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Project Planning</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Project Monitoring and Control</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Supplier Agreement Management</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Measurement and Analysis</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Process and Product Quality Assurance</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Configuration Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Development</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Technical Solution</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Product Integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organizational Process Focus</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Organizational Process Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organizational Training</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Integrated Project Management</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Risk Management</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Decision Analysis and Resolution</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Organizational Process Performance</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Quantitative Project Management</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Organizational Innovation and Deployment</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Causal Analysis and Resolution</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

**Abbreviations**
- SG Specific Goals
- GG Generic Goals
- Amp. Amplifications
- N/A Not Applicable

**Legend**
- Not fulfilled
- Partly fulfilled
- Fulfilled
5.3 Issues spanning multiple process areas

This chapter presents common issues and frustrations with processes and organizational factors encountered during the interviews. These are presented as they were described by the interview subjects and should be considered the subjective opinion and experience of the respondent. It is important to note that this is not an exhaustive list of all issues that were encountered during the interviews. Most issues are covered in the process maturity evaluation in chapter 5.2. The issues described in this chapter are complementary to those described in chapter 5.2 and mainly cover overarching issues that can not be associated with a single process area.

**Product Line A**

This chapter presents a number of issues described by interview subjects from product line A.

**Legacy systems and processes**

According to one respondent, product line A is built around a large number of legacy systems and processes. This, they said, has led to issues across several process areas. Concerning requirements management they state that:

"The simulators that we have are old and no one knows where the requirements that existed back then come from. Everything has just worked and that’s that. Those who developed [the requirements] back then are not with the company any more."

Another respondent added to this that because of these systems age, “they don’t receive a lot of new requirements. So if a new requirement is added it can be a bit loose, so to speak. [...] It’s not that we get a formal document with the requirements.”

One of the respondents also pointed out communication issues and a lack of clear areas of responsibility because they “haven’t done anything new in a long time and you just keep going with what works. [...] It’s very fuzzy because we’ve just kept going like we’ve always done.”

**Lack of documented processes**

The problems encountered when working with legacy systems and processes are according to one respondent compounded by a lack of documented processes. This respondent described how the resulted in issues with knowledge sharing because they have “very many individuals who are all very talented, but it isn’t being disseminated. And with turnovers in staff it has to become more explicit to get it to work.” The same respondent also expressed a lack of documentation in the project planning process:

"The process works well enough, but it’s quite fuzzy what’s supposed to be included and on what level. There are a lot of people involved and the difficult part is to come up with who knows what and who has an interest in this. How does it affect everybody involved? That’s the most difficult. And who decides?"

Similarly, this respondent described how a lack of documentation has led to issues with traceability:

"[The simulator software] has just been updated gradually based on what’s been needed to be done. But traceability and when changes were made, and the strategy behind it all seems to me quite unclear.”

**Product Line B**

This chapter presents a number of issues described by interview subjects from product line B.
5.3. Issues spanning multiple process areas

Unclear Requirements

Multiple respondents expressed frustration with how requirements are gathered and managed. In particular, requirements were described as “being written in a way that is very fuzzy”. The same respondent described how requirements are considered “bothersome to maintain” by the project management team, with the consequence that very little resources are given to maintaining them. This respondent claimed that this was the underlying reason for the reportedly unclear requirements. This opinion was shared somewhat by a different respondent who expressed frustration with the requirements gathering process. According to this respondent, this process often consists of “badgering and badgering [the customer] until coming up with something when you’re already done.”

The unclear nature of the requirements also results in consequences for other parts of the development process according to one of the respondents. According to this respondent, the requirements are often unclear to the point that “it is difficult to deviate from them and it is difficult to achieve them” and that “requirements are not really written in a way that they can either fail or pass.” This respondent expressed some frustration that this resulted in it being difficult to know what they are working towards and making other processes such as testing and verification more difficult due to a lack of clear criteria.

Product integration and external dependencies

The issue of a lack of clear and consistent communication was lifted when discussing the product integration and supplier agreement management process areas. All respondents described issues with how dependencies and deliveries are communicated across teams and projects. One respondent described how the underlying system architecture is built upon modular components but the processes for communicating and coordinating changes to component interfaces are not sufficient:

” [...] there are no avenues of communication at all. This platform and the concept with Gripen, that it’s made of little parts that can be combined is a very good idea. But you still have the old organization with essentially watertight bulkheads. So when [another team] make changes to an interface that breaks the whole operation, that is not done because of ill will but it’s probably because they don’t know that anyone is using this stuff because no one has told them.”

This lack of communication between teams was similarly described by another respondent in the context of prioritizing deliveries by external teams. When asked about the difference in communication with internal versus external teams, they said that “it is a big difference. With your own teams you are in direct contact and can influence their priorities. With external [teams] you don’t really know. All you can do there is to shout ‘Hey, we want something!’”

Lack of quantitative data

One respondent described how a lack of measurable, quantitative data was perceived to be an issue when implementing and evaluating process improvement:

”How do we know that we are improving when we change things? What is the goal and how do we need to improve? I would say there is nothing like that.”

Unclear areas of responsibility

When discussing processes for verification testing, one respondent described how unclear areas of responsibility could be an impediment to the testing process. They described how their team is strictly responsible for testing the simulator platform, but not the hardware and software components used in the actual simulation. According to this respondent, this separation of concerns has led to some “philosophical” debates on testing and areas of responsibility:
"You end up in this discussion: if we have a radar in the airplane, is that not a part of the simulator if it is installed? So there it is extremely difficult to draw the line and I would argue that we don’t have have a firm idea at all on how it should be. The boundaries are very fuzzy."

Product Line C

This chapter presents a number of issues described by interview subjects from product line C.

Lack of defined processes

One respondent described a number of issues related to a lack of defined processes and accompanying documentation. The respondent described how several process areas are characterized more by ad-hoc practices than defined processes. On the topic of requirements management, this respondent stated that “we don’t conduct any structured requirements management” and that “we want to work towards changing this. Because now we see that we’ve been doing this for a long time and it gets very hard to administer and maintain [the requirements]. [...] But then the thing is the individuals who have been working this way for a long time are maybe not used to working any other way.” This lack of documentation was also expressed in the context of project planning:

"[When starting development] it’s more that you’re told ‘we need it by this date’ and then you make a rough plan. But we don’t put together a project directive or project plan like the process stipulates. Here it’s done a bit more ad-hoc."

Similarly on the topic of process and product quality assurance they stated that:

"That process isn’t defined. We have a number of manual tests that can be adapted to new developments fairly easily. But that just manual, there are no automatic tests to ensure quality. Then you sit down with the customer and look at it together."

Overall, when asked about their overall lack of defined processes they stated that:

"I think there are values in this that are worth defending, but I also think that things too often don’t work well enough. We should formalize certain parts a bit more, for instance more explicit gathering of requirements, defining what the work is, to actually build automated tests and to properly verify what we’ve built. Those I would say are the large areas we need to work with."

5.4 Views on PLE

The respondents’ views and subjective opinions of PLE were generally positive. The prevailing opinion was that PLE would be suitable for the type of development work conducted across these product lines. One respondent stated that “I think it would be great, actually. As long as it can be implemented in such a way that it allows for a fairly agile development method. The risk is that it gets process-directed and sluggish, but that isn’t the idea.” Another respondent stated that:

"I naturally feel that we would benefit from it. Of course I can’t really say how much, and that’s partly because of the history and what one has heard before. In part because these are very complex products and it might not be so easy to generalize. What one product requires might not work in another product and then of course it can’t be shared even if the functionality is similar. But that decision
is probably taken somewhat arbitrarily rather than analytically. So that’s why I think we would benefit from PLE and that everyone is aware of that. But it would also require that everyone is aware of what’s on the other side of the fence - what other departments are working on and what they need. [...] That’s where the main part of it lies, that you know what can be used in another product from a technical point of view.”

As can be seen in the quote above, the respondent expressed mostly positive sentiment towards PLE with the caveat that it would require greater communication and transparency between departments. This sentiment was shared by other respondents:

“They way it used to be was that different products were fairly tight, and you might have had a few common components and things that you worked on but everyone really did their own thing. [...] And sometimes you implement some small modification without talking to each other and don’t reuse things. I hope that this new approach will take down some of these walls. [...] I think it’s gonna require more crosswise communication. [...] There are people sitting in the same building and same floor but don’t have any idea that they’re working on the same thing.”

Similarly, another respondent described how aspects of PLE improved their ability to manage the work load but also pointed out that they were skeptical of taking the method “too far”:

“If we didn’t have it we would drown. It would be way too much work. It’s not cost-effective. But you can’t take it too far, either. Then it might get too sluggish. [...] Even if there is a potential benefit it would be so much work to test and verify everything again. Somewhere you have to draw a line for what you should or should not reuse. [...] So you should focus on the right things, some things shouldn’t be shared. But when it comes to working processes and doing things in the same way, there I think we could be more similar.”

Finally, one of the respondents advised caution when putting to much trust in PLE processes:

“There are parts [of PLE] that we surely could adopt and there could be some benefits to gain in those areas. But PLE as a whole would be difficult for our operation since we don’t have a clear end-goal and because we have products which are quite dissimilar. I think a complete PLE implementation would provide a false sense of security in that our products are all alike, when they are actually quite fundamentally different.”

In summary, the respondents were fairly positive towards PLE as way to increase reuse. The main issues and concerns with PLE were mostly concerned with compatibility between PLE and existing processes (e.g. agile development practices) and products (e.g. products being to dissimilar for PLE), and with the organization’s capacity for increased communication and cooperation. However, the respondents that voiced these concerns still pointed out that PLE might still be beneficial since it might drive change, or that aspects of PLE could be useful when applied to a subset of the organization.
This chapter will discuss the findings presented in chapter 5 and evaluate the method used to gather this data. This discussion is grounded in the theory presented in chapter 2 and expanded upon by the author’s own evaluation of the results.

6.1 Results

The analysis of the results presented in chapter 5 can be broken down into three separate avenues of discussion: the derived maturity ratings, the process-related issues that were uncovered, and the subjects’ experiences and opinions of PLE. These topics are covered in the following three chapters.

6.1.1 Maturity ratings

The maturity ratings that were derived as part of this study may be found in chapter 5.2. These will be expanded and reflected upon here in order to provide context for the discussion in the coming chapters.

As it can be plainly seen, all three product lines received a maturity rating of 1: Initial with partial fulfillment of maturity rating 2: Managed. However, there were still fairly major variations in process area fulfillment between the investigated product lines. Product line C in particular had a much lower overall fulfillment ratio. The reasons for this are described in the results chapter. However, despite the lower maturity rating, product line C was described as having values worth defending - implying fairly high process satisfaction. The proposed process improvements were mainly concerned with formalizing certain processes, rather than conducting any sweeping changes or overhauls. This is an important point to keep in mind: despite the proven benefits of moving towards higher CMMI maturity levels [16], this is not always the preferred way forward for the people actually doing the day-to-day work in the organization. In this particular case the theory would imply that moving towards higher maturity ratings could be very beneficial, but the improvements efforts that are actually requested within the organization would do little to bring the product line to the next maturity level. In order to improve the overall process maturity in such an organization, the benefits of moving to a higher maturity rating would need to be effectively sold to the people on the organization if the benefits are not immediately obvious. In the specific case of PLE it is
assumed in the theory that a higher maturity rating implies that the product line is in some way ‘better’\textsuperscript{27}. However, any improvement efforts need to be grounded in the organization’s actual needs and capabilities. Not all organizations need to aim for the highest maturity rating. Rather, any process improvements need to take the particularities of the organization into account and consider these when developing suitable goals\textsuperscript{48}. In the context of this particular case study it is possible that goal should not be to make sweeping changes to product line C to bring it to a higher maturity rating, but rather use CMMI and FEF to guide the process improvements specifically for the areas that were identified as needing improvement. The same can be said for the other two product lines, although they would need improvements to fewer process areas to bring them to the next maturity level.

When analyzing the maturity ratings of the three product lines it is also quite clear that generic goal fulfillment was fairly low across all three product lines. This can be attributed to a general lack of objective data collection and monitoring efforts. As described in chapter\textsuperscript{5.2}, all three product lines generally struggled with data collection. Because generic goal 3 ("Institutionalize a Defined Process") defines a specific practice for process monitoring and objective evaluations, this goal could not be considered fulfilled. Had it not been for this aspect of institutionalization as defined by the CMMI, several process areas would likely have fulfilled the specified generic goals.

Finally, the maturity ratings also provide insight into PLE-specific processes across the three product lines. As the low fulfillment rates show, the investigated product lines have not implemented many of the more mature PLE processes. While this is perhaps not a particularly interesting finding in its own right, it is indeed important to keep this in mind in order to contextualize the issues and experiences described in the coming chapters. The organization described in this case study is not a mature one when it comes to PLE practices. It is however, somewhat more mature when it comes to the more generic CMMI process areas. This is important for understanding the environment in which this study was performed. The organization itself is has existed for quite a long time and fulfills quite a few specific goals for the process areas associated with CMMI maturity levels two and three. However, practices related to data gathering and process evaluation are generally quite weak which manifests in poor generic goal fulfillment. These are the starting conditions for this PLE evaluation and implementation effort which has only just begun. Conscious PLE practices are not yet wide spread and institutionalized across the investigated product lines. This relative immaturity should be kept in mind when reflecting on the coming chapters.

6.1.2 Issues

The respondents described a number of issues that were deemed to be line-agnostic or spanning multiple process areas. These were:

- Legacy systems and processes
- Lack of documented and defined processes
- Unclear requirements
- Product integration and external dependencies
- Lack of quantitative data
- Unclear areas of responsibility

A common thread for several of these issues is a lack of definition, whether that be in form of sub-par process documentation, unclear areas of responsibility or vague requirements. All these things seemingly contribute in their own way to a lower maturity rating. Indeed, defined and well-documented processes are a key component of CMMI, as expressed by the generic goals which aim to institutionalize the organization’s working processes\textsuperscript{9}. Therefore, any
issues that impact this institutionalization process should be dealt with if an organization aims to attain a higher maturity rating. In the particular organization that is the subject of this study we can also identify a thread of legacy systems and processes that is perhaps the underlying reason for this lack of definition. This organization is relatively old and the sort of products they develop are required to be in operation for a very long time. Essentially, they need to develop and maintain the simulators for the entire life time of the plane which can usually span several decades. This means that there is rarely a time for a fresh start or green-field implementations. Instead, legacy systems are necessarily kept and maintained for a long time. What this study found is that with these legacy systems, a legacy culture seems to have developed. This seems to be a culture of "we have always done it this way". For the product lines that were studied this seems to have resulted in processes and knowledge having been concentrated to key personnel that has been with the organization for a long time rather than institutionalized in the organization as a whole. This problem is perhaps exacerbated by the sheer scale of the organization and the seemingly poor communication paths between teams.

This lack of communication between teams is a key issue, particularly considering the context of large-scale simulator construction. These types of industrial simulators are composed of many types of components and crucially of many types of disciplines. The types of simulators described in this study are a complex interplay of hardware, software and simulator models. These three aspects all require different skillsets and were divided across different teams with varying levels of cooperation and communication. Since the construction of a large-scale industrial simulator depends on all three of these groups, clear communication and effective cooperation should be considered necessary for a successful implementation.

A lack of clearly delineated areas of responsibility was also found in this study, which is believed to have exacerbated the problems with cross-team communication and cooperation. Vague responsibilities were described in several contexts but particularly in the product verification process. For the studied organization this led to minor conflicts in the verification phase, where there were disagreements in how to draw the line between the components in the simulator and the underlying simulator system. As it was described, the team developing the simulator are responsible for verifying the base simulator only. Various teams then integrate their respective devices into this simulator platform and are responsible for verifying these devices but not the underlying simulator. While of course some level of separation between these two activities is to be expected, the main point of contention seemed to be that it is not clear where that line is drawn. If it is unclear whose responsibility it is to test and verify a certain component or system, it is reasonable to assume that the quality of that testing activity will suffer. The verification process was also found to be negatively affected by unclear or vague requirements. Requirements were described as being vague and devoid of substance. As previously described, this means that they are difficult to neither fulfill nor fail to meet. This has the effect of them not being as useful as they could be to help design verification tests or guide the implementation.

In a similar vein, the issue of not knowing whom to contact is also closely related to a lack of cross-team communication as well as a lack of defined and documented processes. In the studied organization it was described how a lot of communication between teams is informal and dependent on knowing who to contact on a personal level beforehand. In short, the available avenues of communication are not well-defined or properly documented. This was described as a problem in the project planning process since it means that it is difficult knowing who to contact in order to make a decision. Issues like these can slow down the planning process, wasting resources and time that could be better spent elsewhere. It was also described as a major obstacle in the product integration process. Without proper communication procedures, it was described how changes to public component interfaces had been changed without informing the relevant users since the developers did not even know about the users. This was explicitly attributed to a lack of communication and high barriers between teams by one of the respondents. Especially in the context of PLE such incidents can be very serious. A higher
degree of reuse brings with it stricter demands that changes to these components are made in such a way that dependencies are not affected.

Finally, the issue of a lack of qualitative data can be tied to the low institutionalization rate as described in the previous chapter. By not gathering data for monitoring purposes a large contributing factor to institutionalization is missing. In practice, this lack of data (and monitoring of said data) is expected to lead to process improvement issues, both in terms of identifying improvement needs and in evaluating the effects of improvement efforts. As one respondent expressed it, qualitative data collection provides answers to a key question in process improvement efforts: “How do we know that we are improving when we change things?” Data gathering can also be used to identify the need for improvement and to enable proactive improvement efforts. Especially for a relatively immature organization (PLE-wise) such as the one featured in this study, having the answers to such questions could be very important when implemented the changes necessary to move towards PLE.

6.1.3 Experiences and opinions of PLE

As previously described in chapter 5, the respondents were generally positive towards PLE across all three product lines. Despite the fact that the PLE-related processes were found to be quite immature, the respondents were quick to point to the supposed benefits of PLE in this type of organization. In particular, the general consensus seemed to be that reuse is already a major priority and that PLE is a more-or-less natural extension of that. Since any organizational change process requires the support of the organizational as a whole, seeing that engineers on all levels were generally positive towards PLE is an enabling factor for further PLE initiatives.

There were, however, some concerns and criticism of PLE. This is to be expected for any change process and should be taken seriously into consideration for any organization considering a major process improvement project. In this case, some concerns revolved around the compatibility of PLE and existing processes. For instance, agile development practices were brought up as something that must be accommodated by PLE. Respondents were concerned that the seemingly process-driven and ‘stiff’ nature of PLE would stifle the existing agile practices. However, as argued by Díaz, Pérez, Alarcón, and Garbajosa and O’Leary, McCaffery, Thiel, and Richardson, PLE practices can be successfully combined with agile development methods. As this previous research has shown, the agile practices used in the studied organization could most likely be accommodated by PLE and may even be used to enhance the organization’s ability to deal with unplanned change.

The other main concern with PLE in the studied organization was the fear that it would simply be too time-consuming and might not provide the desired return on investment. This is a valid concern and is indeed one of the most common barriers to PLE adoption. However, as previous research has shown, PLE can lead to lower costs in a surprisingly short amount of time. Krueger has also shown that the proper use of available tools can reduce the implementation effort considerably. In this case, the organization featured in this study would most likely take an extractive approach to PLE adoption since the existing products already have a significant amount of overlap and reuse. This approach can also accommodate a subset of systems being moved over to a product line approach. This means that the PLE effort can be geared towards high-payoff systems in order to minimize costs and maximize payoff. As such, an extractive approach should also address the expressed concern that a full-blown PLE adoption effort would be too restrictive for some of the simulator products.

6.2 Method

Since this is a qualitative case study, the concept of trustworthiness should be considered when evaluating the quality of the research. This is different from the aspects of reliability and validity common applied to quantitative research since the classical meaning of these
6.2. Method

Concepts are not immediately applicable to case study research. This is described in further detail in chapter 2.4.3. The trustworthiness of a case study is instead based on credibility, transferability, dependability and confirmability.

Credibility A major contributing factor for credibility in case study research is obviously the use of established research methods. This study follows many of the guidelines set out by Runeson and Höst and should thus be considered to conform to existing norms and standards for case study research. In particular, the guidelines given by Runeson and Höst are tailored towards research in software engineering, further increasing their applicability. Furthermore, the study took place during a relatively long time (four months, with a two-month active data collection period) during which the researcher worked at the studied organization. This allowed the researcher to gain an understanding of the organizational and cultural context of the organization. However, Shenton mentions that this immersion period carries with it the risk that the researcher’s judgment becomes influenced by the studied organization. While it is impossible to completely eliminate this risk, several strategies were used to prevent bias: a research plan was formulated using previously established research, an interview guide was constructed and used for all interviews and an already established process evaluation method was used. The interviews were also carried out according to guidelines on ethical research and subject anonymity was considered during the entire research process. According to Shenton, this increases the likelihood that respondents are honest during the interviews. Related to this, allowing the subjects to read and comment on or correct the transcript of their respective interviews ensures honesty on part of the researcher and provides the member check suggested by Guba. Finally, the coding process allowed responses to be cross-referenced and validated. While not every subject had detailed knowledge of all process areas, the inclusion of several respondents from the same product line meant that gaps in one respondent’s knowledge could be covered by other respondents. The process of cross-referencing answers across respondents also increases the credibility of the results since it increases the probability of the findings corresponding to reality.

However, a few of the methods for credibility suggested by Shenton were not closely observed. Chief among these were random sampling of participants and triangulation. Randomly selecting individuals for participation in the study was not practically possible, since the subjects needed managerial approval for dedicating the time to participate in the study. Furthermore, participants had to be selected on the basis of availability. All in all, this means that the selection of interview subjects was not truly random but rather driven by practical considerations. Similarly, the reasons for not properly triangulating the gathered data were also practical in nature. The practical hurdles were both a distinct lack of suitable existing third-degree data such as internal documents and a lack of time and resources for collecting other types of first or second-degree data.

The credibility of the study is also plainly related to the extent of data collection and the type of data collected. A major limiting factor for this study is the rather small population: one organization and six interview subjects distributed across three product lines (departments). While this does not discredit the results on its own, it does harm the credibility since it limits the insight that can be gained. However, steps were taken to ensure credibility despite this limitation as described above.

The theoretical framework also affect the credibility of the study, and thus some discussion on the sources used for this work is warranted. In general, PLE is a well-researched topic and has been covered by many different authors and in many different contexts. As such, primary academic sources were typically not difficult to obtain. However, a number of prominent researchers in this field have also independently published books on the topic of PLE. While these should not be considered academic sources in their own right, the background of the authors do lend them some credit. For this thesis, books such as Software product line engineering: Foundations, principles, and techniques. (Pohl, Böckle, and Linden), Software product lines in
action: The best industrial practice in product line engineering. (Linden, Schmid, and Rommes [25]) and Software product lines: practices and patterns (Clements and Northrop [7]) were referenced in order to paint a holistic picture of PLE research. Similarly, textbooks such as Real World Research: A Resource for Social Scientists and Practitioner-Researchers (Robson [37]) were referenced when constructing the research method. When available, these non-academic sources were used to support available academic research. Academic sources were preferred as a rule when available. Taken altogether, this approach is believed to result in high-quality sources and a good overview of available PLE research.

Transferability According to Shenton [46], true transferability can typically not be achieved in quantitative case study research. Instead, transferability is achieved by describing the context of the study and the background of the studied organization. This is done in chapter 3. This background along with the results and accompanying discussion describes the organization and how the results relate to that organizational context. Furthermore, the research plan and process is described in chapter 4 which allows for applicability in process if not in results. However, it is also important to keep in mind that a qualitative case study is a snapshot of a particular organization at a particular point in time and that the study should be understood in this context [46]. This approach allows us to draw certain conclusions about the studied organization and phenomenon, but further studies in different context are needed for more general results. This is a key limiting factor for qualitative case study research and any conclusions must be viewed in light of that context.

Dependability Similarly to how qualitative research is difficult to generalize to different contexts, it is similarly difficult to repeat the results because of the inherently temporal nature of the studied context. As previously described, the research plan and overall method described in chapter 4 allows for replication of the approach but not necessarily of the results. There are close ties to credibility and dependability, with strong credibility going along way to ensure dependability [46]. As such, methods such as triangulation of different types of data would have been useful in ensuring greater dependability. As previously described, such methods were not practical to carry out due to a lack of time and resources. However, as suggested by Shenton [46], the research design and its implementation is described along with the data collection methods are described in chapter 4. They also suggest that the researcher provides a “reflective appraisal of the project”. This is done in chapter 6.

Confirmability Confirmability is concerned with objectivity and is achieved by separating the researcher’s own biases from the findings. To this end, chapter 2 provides an overview of different approaches and chapter 4 describes the reasoning behind the chosen methods. In a similar fashion as with credibility and dependability, triangulating data from different sources would have been useful for ensuring confirmability, but this approach was not used due to the reasons already given. Furthermore, Shenton [46] describes how an audit trail of the steps and decision made during the research process is useful for confirmability. Since this study is rather small in terms of organizations and participants, no major changes or decisions were made to the approach during the research period. The original research plan was followed throughout the entire data collection period and is thus the only step in this audit trail.

6.3 The work in a wider context

By furthering our understanding of PLE in the context of large-scale industrial simulator products, this thesis may make it easier to adopt PLE in such contexts. However, to implement such processes is a major undertaking for any organization and will have a major impact on the people working in the organization. Thus, even if the results of this thesis points towards
PLE being an effective methodology for developing simulator products, any organization considering implementing such methods should take the needs of the business and its people into consideration first and foremost.

PLE has been shown to lead to improved efficiency and quality in software development projects \[35\]. This can have a positive impact on the developing organization as well as customers. It has been shown that PLE can reduce costs and save time in the development process, thus allowing the developing organization to better utilize existing resources. For the customer it has also been shown to increase product quality. By contributing to our knowledge of PLE, this thesis hopefully contributes to these positive aspects as well.

Finally, this study was conducted in accordance to existing ethical standards for case study research as described by Runeson and Höst \[38\]. This means that participants were informed about the purpose of the study and steps were taken to ensure that individual respondents were kept anonymous. Participation was also completely optional and respondents were not pressured to participate. The gathered data was also not reported to its full extent in order to provide anonymity and to ensure that no business or security-sensitive data was leaked outside Saab. This step was also necessary because of the sensitive nature of Saab’s work (military aircraft).
Conclusions

This chapter summarizes the results of the study and relates the findings to the original research question. These conclusions are formed in the basis of the results presented in chapter 5 and the accompanying discussion in chapter 6.

7.1 Research question

The aim of this thesis is to answer the stated research question: To what extent are PLE development practices suitable for developing large-scale simulators? To this end, a case study was conducted where three product lines were studied at the aerospace and defense company Saab. These product lines are part of a very early implementation of PLE, and as expected it was found that PLE-specific processes for these product lines were relatively immature across all three product lines. Because of this limitation, the conclusions in this chapter are based on the theoretical benefits of PLE applied to the studied context.

Opinions of PLE were generally positive across all three product lines. Since reuse is already a major goal and enabling factor in the simulator development process, PLE was generally viewed positively as an effective method for maximizing reuse. Some respondents were concerned about the compatibility of PLE and existing agile development practices. While such concerns are well-founded, previous research has shown that PLE can be compatible with agile practices such as Scrum and that such combined approaches may even lead to an approach that brings with it the best of both worlds.

There were, however, a number of process-related issues brought up by the participants in the study: legacy systems and processes, lack of documented and defined processes, unclear requirements, issues with product integration and external dependencies, lack of quantitative data, and unclear areas of responsibility. When reviewing these issues, we come to the conclusion that a mature PLE implementation would mean improvements to several of these issues. The FEF mandates strong requirements management practices, which should alleviate some of the issues with unclear requirements that were found in the study. Similarly, PLE builds on mature product integration practices and methods for managing dependencies across products. A mature and successful PLE implementation would mean improvements in this process area. The FEF also promotes the use of objective evaluation methods using quantitative data as a tool for process evaluation and improvement. Implementing such practices would both
help implementing further PLE processes and directly address the lack of quantitative data found in this study.

PLE processes could not, however, be considered useful in alleviating the issues of legacy systems and lack of defined processes. Instead, these could be major obstacles to overcome when implementing PLE in the studied organization. In order to implement mature PLE processes, the FEF requires that processes be documented and defined in order to institutionalize these processes within the organization. In the studied organization, many processes are not documented since “they have always been done this way.” In order to implement more mature PLE processes, this culture would need to change in order to drive change.

Some respondents also expressed concern that a full-scale PLE implementation would be too costly and time-consuming. They also stated that some components are not immediately compatible with PLE. To overcome this, a partial extractive approach could be used. This means that only the systems with the highest pay-off would be developed according to PLE.

In summary, this study found that the studied organization could likely benefit from implementing PLE. PLE and the FEF promotes practices that would alleviate some of the major issues found in the studied organization such as unclear requirements, issues with product integration and external dependencies, and a lack of quantitative data. Due to the relative immaturity of PLE processes in the studied organization, these conclusions are based on a review of existing literature and the stated goals and practices of PLE applied to the context of the studied organization.

7.2 Future work

Because of the relative immaturity of PLE processes in the studied organization, the conclusions in the previous chapter are based on the theoretical benefits of PLE. Further research should be conducted when the studied organization has implemented further PLE processes. These studies should evaluate whether the theoretical benefits of PLE found in this study actually led to practical improvements to the stated issues.

Further process-oriented studies should also be conducted on several other organizations developing large-scale simulators in order to identify common themes. While this study took three different product lines into account, they all belonged to the same organization. Studying other organizations would be beneficial for producing more general results.


This appendix describes the amplifications to CMMI which makes up the process dimension of the FEF. The process dimension is largely based on CMMI, with a few PLE-centric amplifications. This appendix lists these amplifications as given by Linden, Schmid, and Rommes [25].

Level 1: Initial

If present, domain engineering, application engineering and collaboration are performed at CMMI level 1.

Level 2: Managed

Domain engineering Performed at CMMI level 2 with the following amplifications:

- **Requirements Management** “manage software product line requirements. Maintain traceability between variation points and variants.” ([25], p. 90)
- **Project Planning** “define variability. Involve application engineering as stakeholder for reusing the domain assets. Define a policy of communication and co-operation with application engineering.” ([25], p. 90)
- **Project Monitoring and Control** “monitor the usage of reusable assets per application.” ([25], p. 90)
- **Measurement and Analysis** “take global product line view into account.” ([25], p. 90)
- **Configuration Management** “pay attention to baseline created and released for reusable assets.” ([25], p. 90)

Application engineering Performed at CMMI level 2 with the following amplifications:

- **Requirements Management** “is management of application requirements, both as reused domain requirements and as application specific requirements.” ([25], p. 90)
Project Planning “reuse domain assets and bind variability. Analyse the risk of dependency on domain engineering. Involve domain engineering as a stakeholder for developing reusable domain assets. Consider the influence of domain engineering on the scope of application projects.” ([25], p. 90)

Project Monitoring and Control “monitor the usage of reusable assets.” ([25], p. 90)

Measurement and Analysis “measure use of common assets by application engineering activities.” ([25], p. 90)

Configuration Management “reusable assets provide a basis for the identification of configuration items.” ([25], p. 90)

Collaboration Performed at CMMI level 2 with the following amplifications:

Requirements Management “maintain bi-directional traceability between software product line and application requirements and use it to identify inconsistencies.” ([25], p. 91)

Project Planning “asset life-cycles live longer than projects. Synchronise between domain and application engineering. Monitor the involvement between domain and application engineering.” ([25], p. 91)

Project Monitoring and Control “monitor and control the synchronisation points between domain and application engineering.” ([25], p. 91)

Configuration Management “change requests regarding application-specific variants of reusable asset variants may lead to change requests on the reusable assets themselves. Synchronise application and domain configuration management.” ([25], p. 91)

Level 3: Defined

Domain engineering Performed at CMMI level 3 with the following amplifications:

Requirements Development “develop requirements for multiple products in a market segment. Define the scope of the software product line. Identify the products to be built. Identify commonality and variability.” ([25], p. 91)

Technical Solution “variability must be included in operational concepts and scenarios for the domain. Develop a platform architecture and the relevant common product derivation support must be defined and implemented. Consider multiple origins and destinations for interfaces.” ([25], p. 91)

Verification “ensure that application engineering makes the proper intended use of domain assets.” ([25], p. 91)

Validation “in application engineering is a stakeholder of the domain validation process.” ([25], p. 91)

Organizational Process Focus and Organizational Process Definition “include the platform for a given domain, procedures of use of this platform, methodologies, reusable components and guidelines. Consider multiple products in a market segment. Use the scope of the software product line.” ([25], p. 91)

Organizational Training “add training on products, application processes and application project groups.” ([25], p. 91)

Application engineering Performed at CMMI level 3 with the following amplifications:

Requirements Development “considers a single customer or market segment. The software product line’s variability and capabilities are used in the communication with the customer. Reuse product line process requirements, bind variability and develop application-specific requirements.” ([25], p. 92)
**Technical Solution** “reuse domain assets, bind variability and develop application-specific assets. Specialise the platform architecture for the application and use the common product derivation support.” ([25], p. 92)

**Validation** “validate both domain and application work products. Staff must be especially trained to know what use they may make of the domain assets. Domain engineering is a stakeholder of the application validation process.” ([25], p. 92)

**Organizational Training** “add training on the platform, asset usage, domain processes and domain project groups.” ([25], p. 92)

**Collaboration** Performed at CMMI level 3 with the following amplifications:

- **Requirements Development** “identify application requirements as potential software product line requirements.” ([25], p. 92)

- **Technical Solution** “determine selection criteria for and co-ordinate the inclusion of application assets in the platform. Communicate existing and planned application and domain assets. Identify application assets as potential domain assets. Co-ordinate make, buy or reuse decisions.” ([25], p. 92)

- **Product Integration** “maintain a roadmap of future products and product enhancements. Determine the actual transfer protocol of deliverables and the timing of the product transfers. Support the integration of domain and application assets.” ([25], p. 92)

- **Verification and Validation** “develop a domain verification environment, procedures and criteria concurrently and iteratively with the application verification environment. Communicate verification results and corrective actions between domain and application engineering. Share a policy of planning between domain and application engineering.” ([25], p. 92)

- **Organizational Process Focus** “determine the organisation’s performance objectives over the whole software product line process. Synchronise action plans between domain and application engineering.” ([25], p. 92)

- **Organizational Process Definition** “assign responsibilities that cover several projects and products.” ([25], p. 92)

- **Integrated Project Management** “communicate existing and planned application and domain assets. Identify application assets as potential domain assets.” ([25], p. 92)

- **Risk Management** “ensure that the risk management strategy and risk mitigation plans cover both domain and application engineering.” ([25], p. 92)

- **Decision Analysis and Resolution** “ensure that alternative solutions’ evaluations cover aspects from both the applications and the domain.” ([25], p. 93)

**Level 4: Quantitatively manged**

**Domain engineering** Performed at CMMI level 4 with the following amplifications:

- **Quantitative Project Management** “integrate the related application engineering sub-processes in the project statistics.” ([25], p. 93)

**Application engineering** Performed at CMMI level 2 with the following amplifications:

- **Quantitative Project Management** “integrate the related domain engineering sub-processes in the project statistics.” ([25], p. 93)

**Collaboration** Performed at CMMI level 2 with the following amplifications:
Quantitative Project Management “measure the dependencies between domain and application engineering and the behaviour of their synchronisation activities. Communicate the influences between domain and application engineering. Negotiate improvement actions on performance of bottleneck projects. Co-ordinate stakeholder identification over application and domain projects.” ([25], p. 93)

Level 5: Optimizing

Domain engineering, application engineering and collaboration are performed at CMMI level 5 and software product family processes are performed at level 4.
This appendix contains the full interview guide used across all product lines. Note that the interviews were conducted in Swedish; as such, this interview guide is also written in Swedish. For a detailed description of interview methodology, please refer to chapter 4.2.

Introduktion

Syfte Syftet med den här studien är att undersöka ett flertal produktlinjer, och utvärdera de arbetsprocesser som används för s.k. ’Product Line Engineering’. För Saabs del kommer det här att ingå i en större utvärderingar av PLE som helhet, och det akademiska värdet ligger i att utvärdera lämpligheten av produktlinjer för att konstrukera storskaliga simulatorer.

Konfidentialitet Det här utförs som en del av mitt exjobb, och det innebär att mina slutsatser kommer publiceras. Den underliggande informationen kommer däremot anonymiseras och granskas innan rapporten sprids externt såväl som intern.

Inspelning Be om tillstånd om att spela in intervjun.

Bakgrund Vad är din roll här på Saab och vad ingår i dina arbetsuppgifter?

Produkt(er) Beskriv produktlinjen och de produkter ni utvecklar. Hur länge har ni arbetat enligt PLE?

Teamet Beskriv teamet.

Vetskap om produktlinjer Är du bekant med konceptet produktlinjer eller ’Product Line Engineering’? Känner du till de produktlinjer som är uppsatta här på Saab?

Information om PLE Beroende på personens kunskap om produktlinjer så kan en kort beskrivning av PLE vara nödvändig.
Processer

Requirements management
SG 1: Manage requirements
- Hur hanterar ni krav för produktlinjen respektive produkter?
- Hur hanteras variationspunkter?
- Hur hanteras avvikelser från kravspec?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva procesen kring kravhantering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring kravhantering?
- Hur utvärderas processen kring kravhantering?

GG 3: Institutionalize a defined process
- Finns det något förbättringsarbete kring kravhantering? Hur bedrivs det?

Project planning
SG 1: Establish estimates
- Hur ser den tidiga planeringsprocessen ut för ett utvecklingsprojekt? Börjar det med en informell uppskattning eller är det ’pang på’ med en fullskalig projektplan direkt?

SG 2: Develop a project plan
- Hur tas en projektplan fram? Risker, resurser, tidsplan etc. Är det en etablerad process eller skiljer sig den överbegränsade processen mellan olika projekt?
- Tänker man på återanvändning och variabilitet redan i planeringsstadiet? Hur komuniceras det mellan olika team/intressenter?

SG 3: Obtain commitment to plan
- Hur skulle du säga att en plan ’förankras’ i organisationen? Är det ett gruppkonsensus, ett fåtal starka ledare som styr med hela handen, informella nätverk, eller kanske något annat?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva procesen kring planering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring planering?
- Hur utvärderas processen kring planering?

Project monitoring and control
SG 1: Monitor against project plan
- Hur mäts/kontrolleras efterlevnad av projektplanen?
- Hur mäts återanvändning av domäntillgångar?
- Hur mäts samarbete mellan domän och applikation?
SG 2: Manage corrective action to closure
- Hur hanteras avvikelser från planer?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva procesen kring projektövervakning? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring projektövervakning?
- Hur utvärderas processen kring projektövervakning?

GG 3: Institutionalize a defined process
- Finns det något förbättringsarbete kring projektövervakning? Hur bedrivs det?

Supplier agreement management
Notera att detta processområde kan hoppas över helt om teamet inte är beroende av externa leverantörer.

SG 1: Establish supplier agreements
- Hur ser upphandlingsprocessen ut för externa leverantörer? Notera att externt i det här sammanhanget är sett från projektteamet - leverantören kan fortfarande vara in-house.

SG 2: Satisfy supplier agreements
- Hur levereras och godkänns externa produkter?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva procesen kring inköp? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring inköp?
- Hur utvärderas processen kring inköp?

GG 3: Institutionalize a defined process
- Finns det något förbättringsarbete kring inköp? Hur bedrivs det?

Measurement and analysis
SG 1: Align measurement and analysis activities
- Sker det någon sort av datainsamling kring efterlevnad eller effektivitet av processer och arbetsmetoder? Det kan exempelvis vara relaterat till kvalitet, resurser, efterlevnad av planer etc.
- Mäts användning av domäntillgångar inom produktlinjen?
- Används datainsamling som beslutstöd?
SG 2: Provide measurement results
- Hur samlas och analyseras insamlad data?
- Hur presenteras data?
- Hur används den för beslutsstöd?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva processen kring datainsamling? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring datainsamling?
- Hur utvärderas processen kring datainsamling?

GG 3: Institutionalize a defined process
- Finns det något förbättringsarbete kring datainsamling? Hur bedrivs det?

Process and product quality assurance
SG 1: Objectively evaluate processes and work products
- Finns det någon process för att utvärdera efterlevnad/effektivitet av arbetsprocesser?
- Finns det någon process för att utvärdera produkter?

SG 2: Provide objective insight
- Hur dokumenteras och hanteras avvikelser?

GG 2: Institutionalize a managed process
- Hur dokumenteras själva utvärderingsprocessen? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring process/produktutvärdering?
- Hur utvärderas processen kring process/produktutvärdering?

GG 3: Institutionalize a defined process
- Finns det något förbättringsarbete kring process/produktutvärdering? Hur bedrivs det?

Configuration management
SG 1: Establish baselines
- Hur identifieras och etableras en produktlinje?
- Hur identifieras produkter som ska konfigurationshanteras?
- Används domäntillgångar som grundplåt för konfigurationshantering?

SG 2: Track and control changes
- Hur spåras ändringar till en konfigurationshanterad produkt?
- Finns det en process för att låta produktspecifika ändringar 'bubbla upp' till domäntillgångarna?
SG 3: Establish integrity

- Sker det någon sorts systematisk gransking av ändringar till konfigurationer? Hur ser den processen ut?

GG 2: Institutionalize a managed process

- Hur dokumenteras själva processen kring konfigurationshantering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring konfigurationshantering?
- Hur utvärderas processen kring konfigurationshantering?

GG 3: Institutionalize a defined process

- Finns det något förbättringsarbete kring konfigurationshantering? Hur bedrivs det?

Requirements development

SG 1: Develop customer requirements

- Hur samlas behov och önskemål från externa intressenter in?
- Hur raffineras kundens behov och önskemål till formella krav?
- Finns det en definierad process för produktlinjen och dess produkter?

SG 2: Develop product requirements

- Hur översätts kundens krav till tekniska produkt- och komponentkrav?
- Finns det en process för att återanvända domänkrav för produktspecifika krav?

SG 3: Analyze and validate requirements

- Hur översätts kraven till konkret funktionalitet? Dvs. hur ser processen ut att översätta produktkrav till konkreta designval?
- Hur säkerställer ni att kraven faktiskt speglar kundens önskemål? Finns det en definierad valideringsprocess?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring kravhantering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring kravhantering?
- Hur utvärderas processen kring kravhantering?

Technical solution

SG 1: Select product-component solutions

- Hur ser den tidiga produktutvecklingsprocessen ut? Hur väljs den ’rätta’ lösningen ut?
- Hur ser förhandlingsprocessen ut mellan alternativa lösningar?
- Tas domänåtgångar i åtanke under den tidiga designprocessen?
SG 2: Develop the design
- Hur ser designprocessen ut?
- Hur tas beslut om variabilitet och återanvändning?

SG 3: Implement the product design
- Hur ser utvecklingsprocessen ut?
- Är det en etablerad process eller skiller det sig från projekt till projekt?
- Hur dokumenteras utvecklingsprocessen?
- Finns det en testningsprocess? Granskning?
- Finns det en kodstandard?
- Utvecklas dokumentation för slutanvändaren parallellt med utvecklingen?
- Hur tas beslut om att tillhandahålla träning eller utbildning för slutanvändare?
- Vem har ansvaret för underhåll av produkten?

GG 3: Institutionalize a defined process
- Hur dokumenteras själva utvecklingsprocessen? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring utvecklingsmetoder?
- Hur utvärderas utvecklingsprocessen?

Product integration
SG 1: Prepare for product integration
- Hur förbered ni er för att integrera komponenter/produkter?
- Finns det etablerade processer för produktintegration?
- Hur långt i förväg brukar produktintegrationer planeras?
- Finns det processer för att stödja integration mellan domän- och produktkomponenter?

SG 2: Ensure interface compatibility
- Hur ser processen ut för garantera kompatibilitet mellan olika produkter och gränssnitt?
- Finns det en formell granskningsprocess för gränssnittbeskrivningar?

SG 3: Assemble product components and deliver the product
- Hur tas beslutet om att integrera två komponenter? Föregås det av en gransknings- eller testprocess?
- Hur sker integrationen rent praktiskt? Är det någon skillnad om en eller flera komponenter kommer från externa team?
- Hur utvärderas den färdiga produkten?
- Hur sker leverans av den färdiga produkten?
GG 3: Institutionalize a defined process

- Hur dokumenteras själva integrationsprocessen? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring produktintegration?
- Hur utvärderas integrationsprocessen?

Verification

Verifikation handlar om att produkten ska uppfylla de etablerade kraven.

SG 1: Prepare for verification

- Vilka kriterier används för produktverifiering?
- Hur bestäms omfattningen av verifikationstestning? Hela produkten eller bara delar?
- Hur ser verifieringsprocessen ut?
- Använder domän- och applikationstillgångar samma verifieringsmiljö och processer?

SG 2: Perform peer reviews

- Hur genomför ni kodgranskning?
- Hur väljs personer ut för att genomföra en granskning?
- Finns det formella kriterier (eller andra dokumenterade resurser) för att genomföra en kodgranskning?
- Vad händer efter kodgranskningen? Vad händer om koden inte godkänns?

SG 3: Verify selected work products

- Finns det en etablerad process för att verifiera att produkten lever upp till kraven?
- Hur analyseras resultaten från verifikationstester?
- Vad händer om produkten inte klarar alla tester?
- Verifieras användningen av domäntillgångar?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva verifikationsprocessen? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring verifikation?
- Hur utvärderas verifikationsprocessen?

Validation

Validering handlar om att produkten uppfyller sitt tänkta syfte i organisationen. Detta är inte nödvändigtvis samma sak som att uppfylla kraven.
SG 1: Prepare for validation
- Vilka kriterier används för produktvalidering?
- Hur bestäms omfattningen av valideringstestning? Hela produkten eller bara delar?
- Hur ser valideringsprocessen ut?
- Använder domän- och applikationstillgångar samma valideringsmiljö och processer?

SG 2: Validate product or product components
- Finns det en etablerad process för att validera att produkten uppfyller sitt tänkta syfte?
- Finns det separata processer för validering av domän- och applikationstillgångar?
- Tar validering av domäntillgångar hänsyn till applikationstillgångar och vice versa?
- Hur analyseras och presenteras valideringsresultat?

GG 3: Institutionalize a defined process
- Hur dokumenteras själva valideringsprocessen? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring validering?
- Hur utvärderas valideringsprocessen?

Organizational process focus
Planera och implementera processförbättringar.

SG 1: Determine Process-Improvement Opportunities
- Har du varit med om några processförbättringsprojekt? Hur gick de till?
- Hur identifierades behovet?
- Hur bestämdes förbättringsförslaget?

SG 2: Plan and Implement Process-Improvement Activities
- Hur planerades processförbättringen?
- Finns det ett systematiskt processförbättringsarbete, eller är det från fall till fall?
- Hur implementerades förbättringen?
- Hur förankrades förbättringen? Vad blev de långsiktiga konsekvenserna?
- Finns det några särskilda processförbättringar för produktlinjer?

GG 3: Institutionalize a defined process
- Hur dokumenteras själva processen kring processförbättring? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring processförbättring?
- Hur utvärderas processen kring processförbättring?
Organizational process definition
Ha en samling processer för hela organisationen. Från dessa grundprocesser kan man sedan göra mer specialiserade projekt specifiera processer.

SG 1: Establish Organizational Process Assets
- Finns det en samling standardprocesser som används i stora delar av organisationen?
- Hur underhålls den samlingen?
- Finns processer för hela produktlivscykeln i samlingen, eller bara för vissa delar?
- Finns PLE-specifika processer?

GG 3: Institutionalize a defined process
- Hur dokumenteras själva processen kring standardprocesser? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring standardprocesser?
- Hur utvärderas processen kring standardprocesser?

Organizational training
SG 1: Establish an Organizational Training Capability
- Finns det rutiner för hur träning eller utbildning ska tillhandahållas?
- Hur bestämmer man om en viss aktivitet eller ett visst arbete behöver speciell träning?
- Finns det någon specifik utbildning kring PLE?

SG 2: Provide Necessary Training
- Hur tillhandahålls träning eller utbildning?
- Vem har ansvaret för att träningen sker?
- Hur utvärderas träningen eller utbildningen?

GG 3: Institutionalize a defined process
- Hur dokumenteras själva processen kring träning och utbildning? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring träning och utbildning? Dvs. vem utbildar lärarna?
- Hur utvärderas processen kring träning och utbildning?

Integrated project management
Driva projekt och hantera intressenter.
SG 1: Use the Project’s Defined Process

- Finns det etablerade processer eller rutiner kring hur ett projekt bedrivs? Kan man säga att projekt följer en viss 'mall'?
- Hur bestämmer man vilka aktiviteter som ingår i ett projekt? Är det baserat på en standardiserad processuppsättning eller är det upp till ledningen för varje projekt?
- Tas produktlinjen och domäntillgångar i hänsyn redan i planeringsstadiet? Är planeringen grundad i produktlinjen?

SG 2: Coordinate and Collaborate with Relevant Stakeholders

- Hur koordineras kommunikationen med alla olika intressenter? Finns det formella kommunikationskanaler eller strategier?
- Hur koordineras beroenden på andra team eller projekt?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring projektledning? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring projektledning?
- Hur utvärderas processen kring projektledning?

Risk management

SG 1: Prepare for Risk Management

- Besrvis det något formellt generellt riskarbete? Alltså inte risker för ett specifikt projekt, utan mer hur ni närmar er riskarbete som en organisation?
- Hur bestämmas det vilka sorts risker som är applicerbara för verksamheten?
- Finns det en formell strategi för riskarbetet?
- Finns det någon koordination mellan riskarbetet för domän- respektive applikation-sutveckling?

SG 2: Identify and Analyze Risks

- Hur identifieras konkreta risker för ett visst projekt eller en viss produkt?
- Hur prioriteras olika risker? Finns det en definierad process?

SG 3: Mitigate risks

- Hur bestämmas strategier för att hantera risker?
- Vem har ansvaret för att motarbeta identifierade risker?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring riskhantering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring riskhantering?
- Hur utvärderas processen kring riskhantering?
Decision analysis and resolution

SG 1: Evaluate Alternatives

- Finns det en formell beslutstagningsprocess för 'viktiga' beslut?
- Finns det riktlinjer för beslutstagnning?
- Finns det riktlinjer för att utvärdera motstridiga lösningar eller alternativ?
- Finns det riktlinjer för utvärderingsmetoder eller beslutsstöd?
- Tar den processen hänsyn till både domän- och applikationsaspekterna?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring beslutshantering? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring beslutshantering?
- Hur utvärderas processen kring beslutshantering?

Organizational process performance

SG 1: Establish performance baselines and models

- Finns det några centrala riktlinjer om hur man ska mäta processeffektivitet? Har ni några sådana rutiner eller riktlinjer?
- Hur mäter ni effektiviteten av en viss process?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring processmätningar? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring processmätningar?
- Hur utvärderas processen kring processmätningar?

Quantitative project management

SG 1: Quantitatively manage the project

- Används kvantitativa mått för att utvärdera och övervaka ett projekt?
- Hur bestäms de kvantitativa måtten?
- Används kvantitativa mått för att utvärdera processer som används i ett projekt?
- Används kvantitativa mått för att övervaka beroenden mellan domän- och applikationsstillgångar?

SG 2: Statistically manage subprocess performance

- Används statistiska mått för att utvärdera och övervaka ett projekt?
- Hur statistiska de kvantitativa måtten?
- Används statistiska mått för att utvärdera processer som används i ett projekt?
GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring kvantitativ projektledning? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring kvantitativ projektledning?
- Hur utvärderas processen kring kvantitativ projektledning?

Organizational innovation and deployment

SG 1: Select improvements

- Hur samlas förbättringsförslag in?
- Hur väljs ett förbättringsförslag?
- Hur utvärderas ett förbättringsförslag?

SG 2: Deploy improvements

- Hur genomförs en förbättring på stor skala?
- Hur mäts effekten av en viss förbättring?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring organisationsförbättring? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring organisationsförbättring?
- Hur utvärderas processen kring organisationsförbättring?

Causal analysis and resolution

SG 1: Determine causes of defects

- Finns det etablerade rutiner för att hitta orsaker till defekter? Ex. root cause analysis.
- Finns det rutiner för datainsamling för att analysera defekter?

SG 2: Address causes of defects

- Finns det etablerade runtiner för hur åtgärder ska tas fram för defekter?
- Hur utvärderas en åtgärd?

GG 3: Institutionalize a defined process

- Hur dokumenteras själva processen kring defektanalys och åtgärder? Finns det en skriftlig process?
- Hur fördelas resurser och träning kring defektanalys och åtgärder?
- Hur utvärderas processen kring defektanalys och åtgärder?
Avslutning

Sammfattning  Sammanfatta kort den övergripande bilden som målats upp under intervjun. Låt den som intervjuas komma med rättelser eller annan input.

Åsikter om PLE  Hur tycker ni att PLE fungerar för er verksamhet? Tycker ni att det har blivit stor skillnad? Har det varit svårt att anpassa sig?