Mälardalen University Press Dissertations
No. 50

KEY ELEMENTS OF SOFTWARE PRODUCT INTEGRATION PROCESSES

Stig Larsson

2007

Department of Computer Science and Electronics

ISSN 1651-4238
ISBN 978-91-85485-61-1
Printed by Arkitektkopia, Västerås, Sweden

Mälardalen University Press Dissertations
No. 50

KEY ELEMENTS OF SOFTWARE PRODUCT INTEGRATION PROCESSES

Stig Larsson

Akademisk avhandling

som för avläggande av Teknologie doktorsexamen i Datavetenskap vid Institutionen för datavetenskap och elektronik kommer att offentligen försvaras torsdagen, 6:e december, 2007, 10.00 i Delta, R-huset, Högskoleplan 1, Västerås.

Fakultetsopponent: professor Paul Grünbacher, Johannes Kepler University Linz, Austria.

Institutionen för datavetenskap och elektronik

Abstract

The product integration is a particularly critical phase of the software product development process as many problems originating from earlier phases become visible in this phase. Problems in product integration result in delays and rework. One of the measures to decrease the late discovery of problems is the use of development standards and guidelines that define practices to ensure correctness of the product integration. However, even if such standards and reference models exist, they are in not used consistently. One of the reasons is a lack of a proof that they indeed improve the integration process, and even more important, that they are sufficient for performing efficient and correct product integration.

The conclusion of the presented research is that the available descriptions in standards and reference models taken one by one are insufficient and must be consolidated to help development organizations improve the product integration process. The research has resulted in a proposed combination of the activities included in the different reference models. This combination has been based on a number of case studies. Through the case studies performed in seven different product development organizations, a relationship between problems that are observed and the failure to follow the recommendations in reference models is identified. The analysis has indicated which practices are necessary, and how other practices support these. The goal with the research is to provide product development organizations with guidelines for how to perform software product integration.

One additional finding of the research is the existence of relation between software architecture and the development process. A method for identifying dependencies between evolvement of software architectures and adaptation of integration practices has been demonstrated.

ISSN 1651-4238

ISBN 978-91-85485-61-1

Abstract

The product integration is a particularly critical phase of the software product development process as many problems originating from earlier phases become visible in this phase. Problems in product integration result in delays and rework. One of the measures to decrease the late discovery of problems is the use of development standards and guidelines that define practices to ensure correctness of the product integration. However, even if such standards and reference models exist, they are in not used consistently. One of the reasons is a lack of a proof that they indeed improve the integration process, and even more important, that they are sufficient for performing efficient and correct product integration.

The conclusion of the presented research is that the available descriptions in standards and reference models taken one by one are insufficient and must be consolidated to help development organizations improve the product integration process. The research has resulted in a proposed combination of the activities included in the different reference models. This combination has been based on a number of case studies. Through the case studies performed in seven different product development organizations, a relationship between problems that are observed and the failure to follow the recommendations in reference models is identified. The analysis has indicated which practices are necessary, and how other practices support these. The goal with the research is to provide product development organizations with guidelines for how to perform software product integration.

One additional finding of the research is the existence of relation between software architecture and the development process. A method for identifying dependencies between evolvement of software architectures and adaptation of integration practices has been demonstrated.

Acknowledgements

Foremost, I would like to express my gratitude towards my supervisor professor Ivica Crnkovic. Ivica has guided me throughout the years, and has been there to help me even when schedules have been overloaded. Many thanks go also to my assistant supervisors Dr. Fredrik Ekdahl, for helping me to start this journey and keeping me on track, to Dr. Rikard Land for all cooperation, and to both of them for interesting discussions on everything from integration and research methods to computer games and langoustes.

Special thanks to Christer Persson, Petri Myllyperkiö, and Stefan Forssander for collaboration on the case studies and for helping me remember the realities of product development, and to Per Branger and Clarens Jonsson for great teamwork and helpful comments. Thanks also to all other colleagues at ABB and all the participants in the case studies.

A major advantage with conducting research studies is all the people you meet. I would like to thank my fellow Ph.D. students Jocke, Johan F., Johan K., Markus, Micke, Peter, and Stefan for reviews, interesting meetings, and guidance to great places.

I would also like to thank all my friends and colleagues at Mälardalen University providing a fruitful environment and giving support when I have needed it.

The work would not have been possible without the support from ABB Corporate Research and KKS, providing me with resources for my research.

I have over the last four years been poor in keeping contact with many of my relatives and friends. Hopefully, I will not start another project like this too soon again so that there will be time to meet (if you still remember me).

Ultimately, this has been possible only through the understanding and support from my wife AnnKi and our daughter Camilla. You are my inspiration and I love you both so much.

Stig Larsson

Shanghai, November, 2007

List of Included Papers

Paper A *On the Expected Synergies between Component-Based Software Engineering and Best Practices in Product Integration*, Stig Larsson, Ivica Crnkovic, Fredrik Ekdahl, Euromicro Conference, IEEE, Rennes, France, August, 2004

Paper B *Case Study: Software Product Integration Practices*, Stig Larsson, Ivica Crnkovic, Product Focused Software Process Improvement: 6th International Conference, PROFES 2005, Springer, Lecture Notes in Computer Science, Volume 3547 / 2005, Oulu, July, 2005

Paper C *Product Integration Improvement Based on Analysis of Build Statistics*, Stig Larsson, Petri Myllyperkiö, Fredrik Ekdahl, presented in a shorter version at ESEC/FSE Conference 2007, Dubrovnik, Croatia, September 2007

Paper D *How to Improve Software Integration*, Stig Larsson, Petri Myllyperkiö, Fredrik Ekdahl, Ivica Crnkovic, submitted to the Information & Software Technology journal, Elsevier

Paper E *Assessing the Influence on Processes when Evolving the Software Architecture*, Stig Larsson, Anders Wall, Peter Wallin, presented at IWPSE 2007, Dubrovnik, Croatia, 2007

List of Related Papers

- *Component-based Development Process and Component Lifecycle*, Ivica Crnkovic, Michel Chaudron, Stig Larsson, International Conference on Software Engineering Advances, ICSEA'06, IEEE, Tahiti, French Polynesia, October, 2006
- *Experience Report: Using Internal CMMI Appraisals to Institutionalize Software Development Performance Improvement*, Fredrik Ekdahl, Stig Larsson, 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), p 216-223, IEEE Computer Society, Cavtat, Croatia, September, 2006
- *Selecting CMMI Appraisal Classes Based on Maturity and Openness*, Stig Larsson, Fredrik Ekdahl, PROFES 2004 - 5th International Conference on Product Focused Software Process Improvement, Springer-Verlag Berlin Heidelberg New York, Kansai Science City, Japan, Editor(s):Frank Bromarius, Hajimu Iida, April, 2004

Additional Publications

Journals

- *Industry Evaluation of the Requirements Abstraction Model*, Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin, in print, Requirements Engineering, 2007.
- *A Model for Technology Transfer in Practice*, Tony Gorschek, Claes Wohlin, Per Garre, Stig Larsson, IEEE Software, vol 23, nr 6, p88-95, IEEE, November, 2006
- *Component-based Development Process and Component Lifecycle*, Ivica Crnkovic, Michel Chaudron, Stig Larsson, Journal of Computing and Information Technology, vol 13, nr 4, p321-327, University Computer Center, Zagreb, November, 2005
- *Integrating Business and Software Development Models*, Christina Wallin, Fredrik Ekdahl, Stig Larsson, IEEE Software, vol 19, nr 6, p28-33, IEEE Computer Society, November, 2002

Thesis

- *Improving Software Product Integration*, Stig Larsson, Licentiate Thesis, Mälardalen University Press, June, 2005

Conferences and workshops

- *Software In-House Integration – Quantified Experiences from Industry*, Rikard Land, Stig Larsson, Ivica Crnkovic, Euromicro Conference, Track on Software Process and Product Improvement (SPPI), IEEE, Cavtat, Croatia, August, 2006
- *Merging In-House Developed Software Systems – A Method for Exploring Alternatives*, Rikard Land, Jan Carlson, Ivica Crnkovic, Stig Larsson, Quality of Software Architecture (QoSA), University of Karlsruhe, Västerås, Sweden, June, 2006

- *Architectural Concerns When Selecting an In-House Integration Strategy – Experiences from Industry*, Rikard Land, Laurens Blankers, Stig Larsson, Ivica Crnkovic, 5th Working IEEE/IFIP Conference on Software architecture, WICSA, p 274-275, IEEE, Pittsburgh, PA, USA, November, 2005
- *Software Systems In-House Integration Strategies: Merge or Retire - Experiences from Industry*, Rikard Land, Laurens Blankers, Stig Larsson, Ivica Crnkovic, Fifth Conference on Software Engineering Research and Practice in Sweden (SERPS), p 21-30, Mälardalen University, Västerås, Sweden, October, 2005
- *Architectural Reuse in Software Systems In-house Integration and Merge – Experiences from Industry*, Rikard Land, Ivica Crnkovic, Stig Larsson, Laurens Blankers, First International Conference on the Quality of Software Architectures (QoSA 2005), Springer Verlag, Erfurt, Germany, September, 2005
- *Process Patterns for Software Systems In-house Integration and Merge – Experiences from Industry*, Rikard Land, Ivica Crnkovic, Stig Larsson, 31st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Track on Software Process and Product Improvement (SPPI), IEEE, Porto, Portugal, August, 2005
- *Concretizing the Vision of a Future Integrated System - Experiences from Industry*, Rikard Land, Ivica Crnkovic, Stig Larsson, 27th International Conference Information Technology Interfaces (ITI), IEEE, Cavtat, Croatia, June, 2005
- *Component-based Development Process and Component Lifecycle*, Ivica Crnkovic, Stig Larsson, Michel Chaudron, 27th International Conference Information Technology Interfaces (ITI), IEEE, Cavtat, Croatia, June, 2005
- *Towards an Efficient and Effective Process for Integration of Component-Based Software Systems*, Stig Larsson, SERPS'03 - Proceedings of the 3rd Conference on Software Engineering Research and Practice in Sweden, Lund, Sweden, October, 2003
- *Are Limited Non-intrusive CMMI-based Appraisals Enough?*, Stig Larsson, Fredrik Ekdahl, Proceedings of the ESEIW 2003 Workshop on Empirical Studies in Software Engineering WSESE 2003, Fraunhofer IRB Verlag, Stuttgart, Germany, September, 2003

- *Combining Models for Business Decisions and Software Development*, Christina Wallin, Stig Larsson, Fredrik Ekdahl, Ivica Crnkovic, Euromicro Conference, IEEE, Dortmund, September, 2002

Table of Contents

Part 1

Chapter 1. Introduction

The product integration process is a set of procedures used to combine components into larger components, subsystems or final products and systems. Product integration enables the organization to observe all important attributes that a product will have; functionality, quality and performance. This is especially true for software systems as the integration is the first occurrence where the full result of the product development effort can be observed. Consequently, the integration activities represent a highly critical part of the product development process.

We refer to the definition of integration for product and system development found in the glossary of EIA 731.1 (interim standard) [1]:

> ”Integration: The merger or combining two or more elements (e.g., components, parts, or configuration items) into a functioning and higher level element with the functional and physical interfaces satisfied. “

This definition describes the product integration process without limiting its use to an implied product development life-cycle model.

Practices for product and system development are described in a number of standards and models such as ISO/IEC 12207 [2] and CMMI [3]. It is noticeable that most standards and reference models deal with product and system development without distinguishing software as a specific item. Steve McConnel describes integration in [4] as “the software development activity in which you combine separate software components into a single system “. Also with this description, it is easy to use the statement (without the software) for any type of integration. However, when going more into detail, there are important differences between the integration of software and other types of integration.

Product integration is in most organizations performed in an iterative and incremental manner, and it is a central part of any product development project. Figure 1 shows a data flow diagram of the product integration process interaction with other processes as described in the CMMI [3]. The

results from design and implementation in Technical Solution are transferred to Product Integration in a controlled manner. The results from Product Integration are used for Verification and Validation activities. When a version of the product or system is ready, it is made available for internal or external customers.

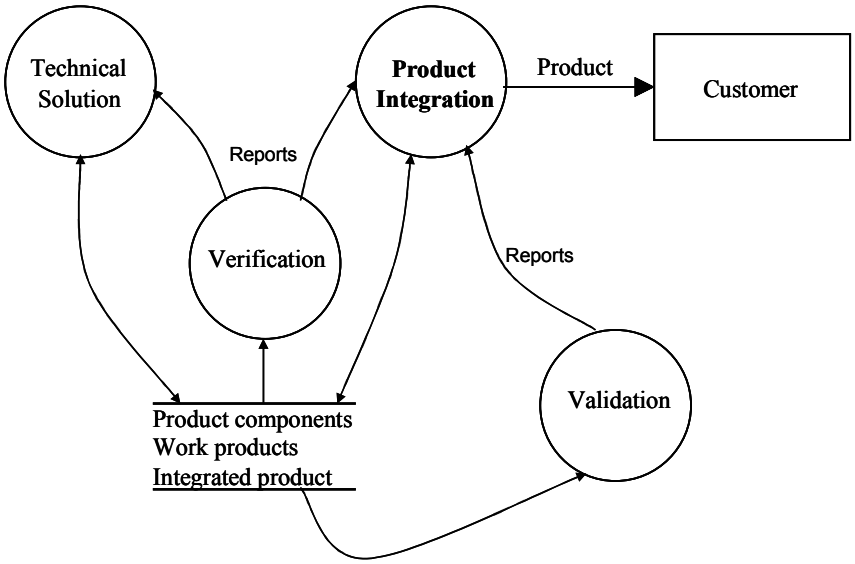

Figure 1. Product integration and related processes

Typical problems in product integration are that the components delivered for integration are not ready, that interfaces between components are insufficiently defined or followed, and that the environment needed for the integration is inappropriately prepared. This leads to the questions if something can be made to improve the product integration process, and what the key elements of product integration are.

1.1 Research Motivation

Good practices for product integration are described in different reference models and standards such as ISO/IEC 12207 [2], CMMI [3], EIA-731.1 [1], and ISO/IEC 15288 [5]. Problems that have been identified include the frequent failure to utilize the knowledge available, or that the recommendations in the reference models are insufficient. This is demonstrated by Campanella who presents an investigation into costs related to different phases in [6], and by RTI describing integration in

relation to testing in [7]. Bajec et al have investigated why available methods are underused in [8] and conclude that inflexibility, and the perceived lack of usefulness are among the reasons. My own experience is that practices described in reference models are too often neglected or misunderstood. This leads to the absence of, inadequate, or insufficient use of activities that would ensure efficient and effective product integration.

There are many examples of how minor mistakes made in an earlier phase complicate and delay the product integration processes. For example, builds fail when code which has not been properly compiled is delivered for integration, or interfaces are changed without checking the impact of that change, or without checking that the corresponding changes in other parts of the system have been done. The consequence is that errors and problems are discovered late in the development process. There are two major negative results from failing product integration processes:

- Activities which use the result and outputs from the product integration process are affected and delayed. These activities include further implementation of functionality, verification, and validation. As integration is performed throughout the project life-cycle, each delay of the results from integration will affect the development effort significantly.
- Work performed in earlier phases must be redone if problems are discovered late in the development process, adding to the needed resources, and further delaying the project results.

Examples from our research include a case study using daily builds showing build statistics that indicate that every fifth build fails due to insufficient use of good practices for product integration. Combining this with the indication that every failed build typically delays the development project by half a day causes a delay in the project of approximately 10% as a direct consequence.

Failure in the integration, which is the result of errors in previous phases can thus be expensive and should be avoided. Practices described in different reference models may help in avoiding these problems. The described practices can be divided into three categories:

- *Preparation of product integration.* This includes decisions on strategy, on integration sequence, and on the criteria for integration
- *Management of interfaces between components.* The integration processes include checking that interfaces are properly defined, and that changes to interfaces are controlled, but not the definition and

design of the interfaces as this is a design issue which is handled in the design process

- *Execution of the product integration.* The execution includes ensuring that the strategy, sequence and criteria are followed, the assembly of components, and the performance of planned tests to verify successful integration

Reference models are of value because they are collected experiences from industry. However, they are not widely used. A question is thus what is needed to help organizations better follow reference models in different product integration undertakings. In addition, the specifics of the reference models differ, and there is a need to understand how these differences may affect the performance of product integration in product development projects.

There is a distinction between products containing software and products that have little or no software: the integration of software products is not tested repeatedly in production through fabrication and manufacturing. This repetition helps the organization prevent problems from reaching the market. This testing is of course most efficient during the development process when the manufacturing organization is active in the project and in the testing to ensure that the production will flow smoothly. Things that can reduce the risk in software product development are the use of continuous or nightly builds, and automated regression testing.

The purpose of this research is thus to determine what changes are required to the current body-of-knowledge for the software product integration process as described in models and standards to be effective.

An additional issue is how the use of the practices described in reference models can be supported in different ways. Examples include training, the use of technologies designed to support product integration, and tools that help engineers define and use components that are well defined are some examples of support that can improve the use of practices described in reference models. Observations made indicate that a closer association between technical and process aspects is needed to ensure the awareness of engineers of the importance of product integration. This means that it is necessary to investigate also the connection and relation between architecture and product integration processes. As a first step, this research considers the influence of architecture on product development processes and proposes a method to find this influence. The use of this method creates

a better understanding of the importance of certain aspects of product integration among engineers.

The importance of architecture in this context is that it affects the possibilities to reach efficient and effective product integration. Product development organizations often have the focus on technical changes, and a wider knowledge of the effects of these changes on the process is needed. If this is developed, we foresee engineers becoming more interested in what affects their work, and how it can be performed more efficiently.

1.2 Research Questions

As described, product integration is a vital part of product development and the main focus and research problem is to understand *what factors influence the possibilities to achieve efficient and effective product integration.* The characteristics of efficient product integration are that unnecessary work is avoided and delays due to integration problems are prevented. Effective product integration is achieved if problems related to the interaction between components are captured, and the planned functionality for a specific integration is achieved. Other important matters related to product integration include delayed time to market, insufficient quality, and inefficient use of resources.

A first step towards understanding how reference models can help was presented in my licentiate thesis [9] and the research presented here is a continued and a more detailed investigation of the reference models. In addition, a first step has been taken in understanding the influence of architecture on processes with emphasis on product integration.

The research questions below make the research topic concrete. The first two questions are related to the use of standards and models as a vehicle for improving product integration, and if there is a need to improve the current reference models. The first question aims to investigate the use of current reference models:

> *Are the practices described in available reference models for product integration necessary and sufficient for visible reduction of problems in the product integration process?* **(Q1a)**

In answering this question, we can find the practices that are most relevant for efficient and effective product integration and what is included in the reference models.

Different reference models cover different aspects of product integration. The reference models represent together the current body-of-knowledge for product integration. Based on the differences and the combined body of knowledge, the next question is:

> *What additions and modifications are needed in the available reference models to take advantage of current body of knowledge in product integration?* **(Q1b)**

Different types of support can help increase use of the described practices. This includes training, tool support, and use of technology that simplifies the product integration.

In addition to the previous question this thesis states a question about integration in a context of software evolution. The reason for this is a recurring observation from the case studies – a relation between changes in the product architecture and a need for changes in the development process. Based on experiences from the case studies we decided to investigate how product development organizations can understand how product integration processes are influenced by changes in the architecture. The evolution of system or product architectures may change the requirements on the product integration process. Failing to change the process when altering the architecture may be one reason why the used product integration practices are not sufficient. We need therefore to understand the influence on process from architectural decisions. This leads to an additional question:

> *How can necessary changes in the integration process due to changes in the product architecture be identified and implemented?* **(Q2)**

1.3 Thesis Overview

The first part of this thesis is an overview of the research, while the second part is a collection of papers that documents details of the research questions, methods, and results.

In part one, chapter 2 relates product integration to different aspects of software product development, including reference models, life-cycles, architecture, product lines, and component-based software engineering.

The method used and the validity of the presented research are discussed in chapter 3, focusing on the whole research project.

Chapter 4 includes a summary of the research results, and an expansion on some of the findings from the papers included.

Chapter 5 wraps up the overview part with conclusions and a look at possible future work.

Part two of the thesis includes the following papers:

Paper A "On the Expected Synergies between Component-Based Software Engineering and Best Practices in Product Integration"

This paper describes the product integration practices in one product development organization. Problems observed are compared with component-based development practices to investigate if these can help the organization follow good practices as described in the CMMI.

Presented at the Euromicro Conference, Rennes, France, August 2004. Authors: Stig Larsson, Ivica Crnkovic, Fredrik Ekdahl.[10]

I was the main author; I contributed with the description of good practices in product integration, the methodology, the case study, the analysis and conclusions. The co-authors contributed with advice regarding methodology, discussions regarding the analysis and conclusions, and reviews.

Paper B **"**Case Study: Software Product Integration Practices"

This paper includes case studies from three organizations. Practices used in the organizations are compared to EIA-731, and the problems encountered by each of the organizations are described. Problems are mapped to practices, and the conclusion is

that the standard includes activities that can help organizations avoid problems which can appear when integrating components to systems.

Presented at PROFES 2005 Conference, Oulu, Finland June 2005. Authors: Stig Larsson, Ivica Crnkovic. [11]

I was the main author; I contributed with the description of good practices in product integration, the methodology, the case study, the analysis and conclusions. The co-author contributed with advice regarding methodology, discussions regarding the analysis and conclusions, and reviews.

Paper C "Product Integration Improvement Based on Analysis of Build Statistics"

This paper proposes a method for mapping project data to different practices and combines this mapping with project appraisal results to form a basis for focused performance improvement. The product integration processes in four projects from three organizations were examined using the proposed method and the findings are presented. The study demonstrates how the two components, collected metrics and appraisal results, complement each other in the effort to develop product integration process improvement effectiveness.

Presented in a shorter version at ESEC/FSE Conference 2007. Authors: Stig Larsson, Petri Myllyperkiö, Fredrik Ekdahl. [12]

I contributed with the description of good practices in product integration, methodology, and two of the case studies and the analysis for these as well as the conclusions. Petri Myllyperkiö contributed with two case studies, and for these we made the analysis together. Both co-authors contributed through discussions and reviews.

Paper D "How to Improve Software Integration"

This paper consolidates the investigations in paper A, B and C with chapter 4 of my licentiate thesis [9] to show the possibility to enhance current reference models. Seven case studies are compared to five reference models. A combination of the findings from the cases and the models result in a proposed set of 15 practices for successful product integration.

Submitted to the Information & Software Technology journal, Elsevier. Authors: Stig Larsson, Petri Myllyperkiö, Fredrik Ekdahl, Ivica Crnkovic [13].

I contributed with the description of product integration practices in reference models, methodology, and five of the seven the case studies. I also made the analysis which was then discussed with the co-authors, and prepared the conclusion. All co-authors contributed through reviewing the paper.

Paper E "Assessing the Influence on Processes when Evolving the Software Architecture",

This paper expresses different relationships between architectural changes, process changes and the underlying business objectives. As an example of how the understanding of these relationships can be used, we describe a method for assessing the process changes needed when refactoring is performed. Details regarding the consequences for the product integration process are included as examples.

Presented at the 9th International Workshop on Principles of Software Evolution, IWPSE, 2007. Authors: Stig Larsson, Anders Wall, Peter Wallin.[14]

I was the main author and lead the study. I contributed with the description of the proposed method, while the case description, the related work, and the conclusions were made in cooperation with the co-authors.

In addition, the following papers are indirectly related to the thesis. Material from these papers has been used in the preparation of part 1 of this thesis:

- "Component-based Development Process and Component Lifecycle", Ivica Crnkovic, Michel Chaudron, Stig Larsson, International Conference on Software Engineering Advances, ICSEA'06, IEEE, Tahiti, French Polynesia, October, 2006 [15]
- "Experience Report: Using Internal CMMI Appraisals to Institutionalize Software Development Performance Improvement", Fredrik Ekdahl, Stig Larsson, 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), p 216-223, IEEE Computer Society, Cavtat, Croatia, September, 2006 [16]

- “Selecting CMMI Appraisal Classes Based on Maturity and Openness”, Stig Larsson, Fredrik Ekdahl, PROFES 2004 - 5th International Conference on Product Focused Software Process Improvement, Springer-Verlag Berlin Heidelberg New York, Kansai Science City, Japan, Editor(s):Frank Bromarius, Hajimu Iida, April, 2004 [17]

Chapter 2. Product Integration

This chapter describes product integration, beginning with a general discussion about different interpretations of the nature of product integration. With this background, the problems found in product integration, as used in this thesis, are described. In addition, we discuss the use of reference models, as well as other concepts in software engineering related to and affecting product integration processes.

2.1 Interpretation of "Product Integration"

The terms "product integration", "systems integration" and "software system integration" are used for several different aspects in product and system development literature. Grady claims that *integration* is one of the most misunderstood concepts within systems engineering [18]. Djavanshir and Khorramshahgol [19] have investigated the importance of different process areas related to system integration and observe that professionals in the field relate integration to many areas of systems engineering. This indicates that there is no clear definition of integration when discussing system and software engineering. It is consequently necessary to clearly define the scope of integration, and to be aware of other interpretations of the term. Sage and Lynch provides an overview in [20], and Land elaborates on different meanings of the terms in [21]. The main uses of the terms are:

- Product integration processes:
  This term describes the process used in product development projects when parts are combined into more complex parts and eventually into the product or system to be delivered to the customer. It includes the activities ensuring that the combinations of components are functioning as intended and the management of interfaces between components and between the product and the environment. As earlier described, this is the focus for this thesis.

- Architectural, or technical, product or system integration:
  This concerns the technical solutions used to fulfill requirements on functionality and quality attributes such as reliability and performance. Different levels of integration include export and import facilities, the use of wrappers and adapters, integration through shared databases, and integration on source code level. Interface design is one important issue for all levels of architectural integration, and standard interfaces are available for many applications. Different types of architectural integration is described by Nilsson et al in [22]. Other examples of the use of integration in this meaning is found in [23] where Garlan describes trends in software architecture research, and in [24] where Gorton describes useful architectural practices .
- Enterprise Application Integration (EAI)
  EAI is a specific type of architectural integration where organizations combine and integrate existing and new systems to assist the organization in achieving business objectives. This type of integration is performed to ensure data consistency and to make information accessible to different types of stakeholders, often based on the use of a common middleware. Examples of descriptions of EIA are [25] by Cummins, [26] by Linthicum, and [27] by Ruh et al.
- Software system in-house integration:
  When merging systems with similar purposes, there are both process, architectural, and technical considerations to be managed. This has been described by Land in [21].
- Integrated product and process development:
  The integration of product and process development aims at having a focus on collaboration between all stakeholders in the product development. An emphasis is put on a common vision which is key to fulfill and exceed customer satisfaction. This includes all different disciplines needed to work together in a common effort, often as one project, throughout the project life-cycle. The development processes proceed in an integrated project in parallel, which requires tight cooperation between the participants. The use of integrated product and process development is included in the CMMI [3], and has for example been described by Parsaei et al in [28]

2.2 Product Integration Problems in the Industry

To understand what needs to be improved in descriptions and implementation of product integration processes, it is important to understand what types of problems are found in industry.

Problems in product integration have been described by Ramamoorthy in [29]. According to that study the software system integration problem includes several issues:

- Inconsistencies in the interfaces between modules in the system lead to problems at integration time. The inconsistencies result from the different assumptions made by engineers in earlier phases of the development
- Insufficient use of strategies and planning for the integration effort. This leads to unnecessary dependencies in the product integration for the different modules, and to increased need for interactions between designers to synchronize deliveries
- Insufficient understanding of the dependency structure of the product or system leads to cumbersome debugging and fault finding at integration time

Through the case studies performed in this research project, we have been able to observe a more detailed view of the problems and the following types have been found:

- Related to architecture and design
  - Architectural decisions are done without considering the full system, leading to problems at integration time
  - Changes are made to interfaces without proper control. This leads to errors in the builds or initial integration testing
  - Changes in common resources (e.g. common include files) are not controlled. This results in errors appearing in other components which have not been changed
  - New functions are added and errors are corrected without proper investigation of consequences. The result may be new errors that influence the functionality and performance of the system more than the original problem
  - Errors appear in other components which have not been changed due to changes in interfaces, i.e. changes are made in how two

components interact, while also other components are using this interface

- Related to the inadequate establishment or use of the integration environment
  - Problems appear as tests for the components are not run in the same type of environment as the integration test system. Different versions of hardware and test platform are used
  - The build environment is not prepared for new builds, e.g. results from earlier builds are not removed before a new generation of the system is started
  - Untested changes are introduced in the integration environment e.g. build scripts are changed without proper verification
- Related to inadequate delivery of functions
  - Inconsistent code, i.e. functions that have only been partly implemented, is delivered for integration. Files are not included in the build as planned, resulting in failed builds
  - Functions are not always delivered in time for integration or may be incompletely delivered. This leads to problems in the build process or in integration and system tests
  - Functions are not always fully tested when delivered for integration. This leads to problems in the build process or in integration and system tests

The types of problems are not independent. An example of this is that inadequate coordination of when different components are to be delivered may lead to pressure to deliver components without proper preparation or testing. If no agreed criteria have been defined, it will be even easier to accept this behavior.

To summarize, the problems are in essence related to interaction and planning for interaction, both between different development teams and between the components that are to constitute the final product.

Through the studies performed in industry we have seen that the investigated systems all have some type of legacy. This is an additional factor that creates limitations for how to perform product integration. The legacy can be an inherited code base, connections to other systems such as tools that require certain components to ensure backward compatibility, or standards that require specific behavior from the system. The result of this is

that the product integration depends on a large number of earlier decisions and resulting strategies. In turn, the consequence of this is reduced freedom to select strategy for the integration, and may lead to needs for refactoring or other changes to the architecture before a new strategy can be selected.

2.3 Applying Reference Models

The reference models used in our research describe and propose different activities that should help in achieving efficient and effective product integration.

Two types of reference material, standards and models, have been considered in this study and are referred to as reference models[1]. Product integration is treated in different ways in the reference models; in some models such as ISO/IEC 12207 [2] and CMMI [3], the subject is handled in a specific part, while in others such as EIA-632 [30], the description of product integration is found in different sections. In most reference models, the product (or system) integration is considered to result in aggregations of components into bigger components. The product integration is repeated over the project life-cycle until the product or system is available and can be delivered to the customer.

The activities that are considered part of product integration can be divided into three areas: preparation, management of interfaces, and execution of the product integration.

Careful *preparation* is in the reference models described as the key to efficient and effective product integration. It includes defining a strategy based on business needs and targets, and organizing the integration sequence to be in accordance with the strategy and synchronized with other project and organization activities. An environment for the integration should be prepared, and requirements on the components to be delivered to integration defined.

Many system and product integration problems occur due to incomplete or misunderstood interfaces. Therefore *management of interfaces*, i.e. the identification and definition of what interfaces should be managed, need to

[1] The difference between the types is that standards have been approved by a standardization body, while a model may be issued by any company or organization.

be a part of the product integration. However, the design and implementation of interfaces should be considered part of the architectural and detailed design. The target for the management of interfaces is to ensure compatibility. This means that the practices needed is (i) to review the interfaces for completeness, and (ii) to manage relationships between interfaces and components to ensure that any changes to interfaces are dealt with in affected components.

When using the reference models as a basis for implementing management of interfaces, it is important that the concept of interfaces is clearly understood. Interfaces are not only the syntactical description of the connection point to a software component. An example of how this is captured in the reference models can be seen in ISO/IEC 15288 [5] where section 5.5.4.3 includes the following

> ”g) Define and document the interfaces between system elements and at the system boundary with external systems
>
> Note Definitions are made with a level of detail and control appropriate to the creation, use and evolution of the system entity and with interface documentation from parties responsible for external interfacing entities. …”

This specific standard covers all types of engineering, and the statement needs to be complemented with more details of what should be considered as a part of interfaces to be practically useful when developing software intensive systems. In addition, each organization needs to determine what is needed. One view of how to regard software interfaces is given by Parnas in [31]. He describes how interfaces need to comprise *the set of assumptions that the developers of the different components can make about other components*, e.g. the behavior in normal and error situations, resource needs, and the need for other components.

The *execution* takes advantage of the preparations, and includes checks that the criteria for when components can be delivered are fulfilled, that the components are delivered as planned, and the integration including evaluation and test of the assembled components.

A detailed description of where information about the three areas *preparation*, *management of interfaces*, and *execution* can be found for different reference models is available in paper D [13].

In [32], Stavridou investigates integration standards for critical software intensive systems. The examination focuses on military policies and standards, but includes also ISO/IEC 12207 in the comparison. The

conclusion is that the majority of the examined standards address integration testing, but that the standardization is not appropriate for many integration issues. The descriptions of the included activities are insufficient as support for a project manager running a product development project. An additional conclusion is that the integration activities should be considered as a separate phase of system development.

Incorrect use of reference models and software development models is described by Fitzgerald [33]. The reason for not using the models as intended is claimed to be the perceived lack of contribution to successful product development, and inflexibility in the models, not allowing for customization to specific organizational and project needs. This has also been highlighted by Bajec et al in [8]. They describe and prescribe a method for adapting the development model to the specific project. This should of course also include the product integration part of the process.

One reason for the industry to be slow in adopting useful practices from reference models may be their format and their content. Reference models in general cover projects and organizations with a large range of attributes; projects with significant differences in size, distribution, complexity and novelty should be covered in the same models. This means that the models often describe *what* should be performed, but not necessarily *how*. The interpretation of a specific method or practice becomes important, and the insufficient knowledge in how to implement the practices may prevent the organizations from adhering to a model.

The extent to which the models describe how a practice should be implemented differs. However, none of the models used in our research are explicit and give detailed advice on how the models can be used for different types of projects and organizations. Most detailed is the CMMI [3] which describes subpractices and expected work products, while standards such as ISO/IEC 12207 [2] give only high level direction.

2.4 Product Integration and Related Software Engineering Concepts

Several areas related to product integration have been identified in literature. That the areas are related to product integration has been confirmed in the examination of the organizations and projects that form the basis for the research presented in this thesis.

Two basic topics are how the selection of the project and software lifecycle influences the product integration, and the effect architectural decisions have on product integration. Other areas such as distributed development and the use of the software product line concept are phenomena that make the product integration more complex.

In this section, a set of these software engineering concepts are discussed from the perspective of product integration. The selection has been made based on literature search and investigations of areas covered in major software engineering conferences.

2.4.1 Project Life-cycle Models

McConnel stresses the importance of selecting the life-cycle model that is appropriate for a specific type of development and provides a selection guide in [34]. The selection of the project life-cycle model also determines what options the project will have from which to select integration strategy. In [35], Pressman differentiates between three types of models: the waterfall model, incremental models, and evolutionary models. Each of these types has an influence on how the product integration processes can be implemented.

The *waterfall model* requires that each phase of the development is concluded before the next is started. A strict use of this model will force the project to begin the integration when all components are ready and apply a big-bang approach. However, the strategies and sequences for integration can be selected on the basis of the needs from the organization if the schedule permits. This includes incrementally integrating components based on architectural or other considerations even though all components are available. There is a risk that errors found in integration requires the project to modify components or interfaces which will delay the project. Modifications to the model permit overlapping phases, enabling the organization to select integration sequence by giving priority to which components should be ready first. Also, by applying the model separately on different components and subsystem, a more flexible integration process can be implemented. The waterfall model used in this way resembles an incremental model.

Using *incremental models* increases the number of possible strategies for product integration. The selection of integration strategy of may determine what should be developed in each increment. Considerations for the project planning with regards to increments will resemble the considerations for the integration selection. Examples of different strategies are to provide the

basic functionality in the first increment, and making more advanced versions of each feature available as the project proceeds with further increments, or to develop the most important feature with all functionality available for the first increment. One important aspect for the product integration is that a strategy and integration sequence is also needed within an increment, e.g. a specific order is needed to be able to perform the integration tests.

Evolutionary models include two major types: spiral models and evolutionary prototyping. Spiral models have been described by Boehm in [36] and further developed in [37], and focus on minimizing risk by starting the project in small scale, addressing the major risks. The project then iterates a number of steps, including setting a target for the iteration, identifying risks, evaluating alternatives, developing deliverables as described for the iteration and evaluating the results, planning the next iteration and deciding on an approach for the next iteration. The integration process will in most evolutionary projects differ between the iterations based on the approach and purpose of the specific iteration. This also gives the organization and project the option to adapt the product integration as the project proceeds.

Evolutionary prototyping is also iterative, and focuses on aspects of the product that can be evaluated by the customer (or a representative for a customer base). Quick design and implementation lead to early feedback which can be used for refining the requirements. Later versions of the product will be designed with more focus on architecture and quality. Here, the product integration processes will be very important, especially for the handling of interfaces. Early prototypes tend to be built on existing components that may have to be replaced in a later iteration, and the management of interfaces and changes to these is crucial.

Using evolutionary models put higher demands on the project as the focus is on minimizing risks and as the processes are often adapted as the project progresses.

Ramamoorthy presents a proposal how to tackle the challenges in product integration in [29], and relates the activities to different project life-cycle phases. The proposal resembles the activities described in reference models, and give additional views on what can be used as design guidelines when implementing product integration processes. The proposal includes a *preliminary development phase* which incorporates specification of interfaces, establishment of an integration strategy, the implementation of an

integration environment, establishment of criteria for integration, and the development of an integration plan. The second phase is labeled *initial integration* and includes primarily management tasks. Examples of activities are regular feedback on status, improvement of the strategy, and monitoring the integration process. The *final product integration* phase is briefly described. Similar to the reference models, no validation of the proposed method or the included activities is presented.

The activities in the product integration area have also been the subject of interest from the agile community where continuous integration is common and frequent builds is one of the cornerstones. One example is [38] where Fowler describes the requirements on developers: before committing components back to the mainline the developer would need to update his work area with the latest mainline, i.e. build against the latest changes of other developers. Only after that, integration into the mainline would be permitted. The use of continuous integration and frequent builds is one of the strategies that can be selected for product integration, and will also put requirements on other activities such as the preparation of an integration environment.

2.4.2 Architecture and Product Integration

Architecture and design are connected to the product integration processes in several ways. The interface design affects the possibilities to select different integration strategies, while the chosen integration strategy may influence and limit the architectural options available. That management of interfaces is an important aspect of many of the architectural tactics as described by Bass et al in [39].

Sage and Lynch provides a general description of system and product integration in [20] and describes a view of how the integration can affected by architecture. The conclusion is that developing an appropriate architecture for a system will simplify the integration later in the project's lifecycle. It is also stated that the architecture can be the means for communication and knowledge transfer in a project. This is further described by Ovaska et al in [40]. The main idea in the description is that a common understanding of the software architecture between the software development parties will improve the coordination of different teams. They also stress the need for both informal communication and formal descriptions of interfaces.

Eppinger describes in [41] a method to reduced the problems in integration using an architectural and design structure matrix approach. The method

includes three steps: decomposition, identification of interactions between the components based on different types of interaction, and clustering of components based on the analysis of the structure of interactions. The method is closely related to the management of interfaces as described in product integration.

Another area that has been well researched is how software can be reused. One example in the context of architecture and refactoring has been described by Metha and Heinemann in [42] where an evolution model is proposed and a methodology that finds code that can be refactored into components is described. Chioch et al [43] reports on experiences where the process determine the acceptance for an architecture intended for reuse.

The challenge of integrating large systems is discussed by Schulte in [44], who proposes methods for modeling system behavior to handle the uncertainties in resulting system characteristics when integrating components. According to Schulte, three areas need to evolve to provide capabilities powerful enough to assist when modeling real-time systems. These are multi-view modeling, analysis and code generation. Another example of using models to ensure efficient and effective integration has been presented by Karsai et al [45]. The point made is that modeling should be made the central activity when developing systems.

The focus of the research presented here is on embedded industrial systems with specific requirements on different quality attributes such as timeliness, reliability, and availability. This is reflected in the need for specific approaches both regarding the view of computation as described by Lee in [46], and in the fact that in these systems, physical properties are modeled and appear as cross-cutting constraints for the whole system as described by Sztipanovits and Karsai in [47]. To solve these needs and requirements, appropriate architectural solutions will be necessary.

Also with respect to the binding time[2], there are requirements that will influence the product integration. Svanberg et al describes the concept of binding time in the context of product lines in [48]. A distinction is made between pre-delivery binding time which includes product architecture derivation, compiling and linking, and post-delivery binding time which can be at start-up, during runtime, or per call. One characteristic of embedded industrial systems is that most bindings are performed pre-delivery, and that

[2] Binding time is the moment when a decision is made for a possible variation in the product.

binding at start-up primarily is performed through configuration. Binding per call, during run-time, is rarely used in this kind of systems.

2.4.3 Software Product Lines

Software product line engineering is a technique used to utilize a common set of core assets in the development and preparation of a series of products. This concept requires a different approach than traditional software product development. Core asset development and their utilization to build products need planning, and require efforts. This includes strategies that encompass several products, management that enforces the development and utilization of common assets, and technologies, methods and processes adapted to software product line development. Bass et al observes that the use of product lines will replace design and coding with integration and testing as the predominant activities in [39].

The software product line concept is described in detail by Clements and Northrop in [49], and by Pohl et al in [50]. SEI provides additional information, references and examples on the "Software Product Lines Home Page" [51].

SEI describes the particular aspects of system and product integration in [52]. One specific topic which differs from One-off product development is the pre-integration that is made on the core assets. This pre-integration has two purposes. The first is a verification to ensure that components that are part of the core assets can be integrated as intended. The second purpose is to prepare larger components that can be reused. This is done to reduce the effort when instantiating products.

Some recent research describes additional advancements. In [53], Krueger describes three new methods which can increase the usefulness of software product lines. These are "Software Mass Customization", "Minimally Invasive Transitions", and "Bounded Combinatorics". The first method can affect the product integration process and to a large degree reduce the effort for integration. It builds on the concept that a software product line (SPL) configurator uses predefined product definitions to create product instances. Besides reducing the need for application development, the recreation of products when changing core assets can be automated. Using an SPL configurator also changes the organizational needs: the development will be performed on the core assets that will contain all software necessary for all the product instantiations. However, there is still a need to exercise the practices for product integration when developing and verifying the core assets. Additional activities include decisions on variation points, and

verification strategies for these. The third method, "Bounded Combinatorics", reduces the variations and resembles the pre-integration technique described by SEI in [52].

2.4.4 Distributed Development

Distribution of development efforts in a project increases the need for monitoring communication between the participants in the project, and also with other stakeholders. The general considerations has for example been examined by Herbsleb and Mockus in [54], and Paulish et al in [55]. Conclusions from these studies show that distributed development takes more time and requires more effort that single-site development. The investigations also provide guidance on how to minimize the negative effects, emphasizing communication on all aspects of the development. Paulish et al note that the architectural work primarily was performed in face-to-face meetings and workshops, focusing on specific topics in each meeting. Interface design and communication regarding content of builds were considered very difficult.

This is also described by Vand den Bulte and Moenaert in [56] where they show that organizational boundaries, and especially physical distance, may hinder communication between distributed development teams, especially for technical know-how.

Sosa et al combine the perspectives of product architecture and organizational structure in [57], and investigate the communication patterns in organizations based on interfaces described in the product architecture. The three main findings are that misalignment of interfaces is greater across organizational and system boundaries, that indirect interaction is important to achieve coordination, and that modularization may hinder alignment of interfaces and interactions. All three findings support the need for careful management of interfaces which is one of the main themes of product integration.

Komi-Sirivö and Tihinen present an investigation into the factors which determines the success or otherwise of distributed development in [58] and lists interfaces as being the most important source of software errors after misinterpreted, changing, or missing requirements. This highlights the importance of interface management in distributed environments.

Product integration is affected by distribution of development efforts as the management of dependencies both between development activities and between parts of the system becomes more cumbersome. This underlines the

importance of the three areas covered in reference models for product integration: 1) preparation, to get a common vision and agree on plans, environments, etc, 2) interface management, to ensure that information that affects components developed in different locations is communicated in an efficient and effective way, and 3) execution, monitoring the cooperation as the project proceeds.

2.4.5 Component Based Software Engineering and Development

Component based software engineering may be one tool in improving the engineering practices and simplify product integration practices. However, there are indications described by Crnkovic in [59] that changes are needed in the established development and life cycle models. The differences between component-based development and non-component based development require the use of new patterns, and a distinction between development *of* components and development *with* components. The product integration process is one area in which we can anticipate changes in current practices as the use of general-purpose components for product development of embedded systems is increasing. There are several definitions of a software component is in this context, and in this section we use the focused definition by Heineman and Councill found in [60]:

> "A *software component* is a software element that conforms to a component model and can be independently deployed and composed with modification according to a composition standard.
>
> A *component model* defines specific interaction and composition standards. A *component model implementation* is the dedicated set of executable software elements required to support the execution of components that conform to the model.
>
> A *software component infrastructure* is a set of interacting software components designed to ensure that a software system or subsystem constructed using those components and interfaces will satisfy clearly defined performance specifications."

There are several approaches to architecting and implementing component based development (CBD). Dogru and Tanik describe in [61] a fully component-oriented approach and contrasts this with modifying object oriented approaches, stressing that CBD takes no account of inheritance and capitalizes on composition. Van Ommering describes in [62] a component model that is used as the basis for development of product families in a

distributed environment. One interesting part of this description is how the process and organization have been aligned with the new way of developing products. This example describes specific changes in the organization; the division into an asset team, handling the basic system, and product teams that build the applications on top of the system and integrates the final product. The conclusion arrived at, with respect to process, was the increased importance of the role of the "quality officers" ensuring that the standards for the specific development methods were followed.

The differences between the development of component based systems and non-component based systems are described in [63]. The separation of component development and development of systems based on components is highlighted and this description gives input to how integration can be organized. Morisio et al [64] also describe the difference in the development of components and system in the context of COTS (commercial off-the-self) components. The investigation included fifteen projects, and the integration was, for many of the investigated projects, the activity that consumed most effort. A solution by means of which decisions regarding requirements and candidate components are made together is proposed. The method also includes early analysis of integration issues in the design phase.

The importance of following and performing all process steps is described by Tran et al in [65]. Their investigation shows an increased risk of failure if a project omits any of the defined steps: identification, selection, evaluation, procurement, integration, and evaluation of software components.

de Jonge finds that the goals of reusing building blocks and the goals of integration are difficult to combine, but proposes techniques of how this can be done [66]. These include the concept of source code components and source tree composition that integrates source files, build and configuration processes.

One area that is related is the use of generic component architectures. In [67], Lichota et al describes a generic component architecture and proposes a process for selecting and integrating software products as components. This process is described as five steps:

- **Identification:**
  Determination if a candidate component can be considered to be included in the product.
- **Screening:**
  Components to be further investigated are selected on the basis of a

review of all available information about the components selected in the identification phase.

- **Stand-Alone Test:**
  Each component is tested to determine if it fulfills the expectations described in the documentation. In addition, the components should be tested to determine its potential reliability, reusability, and general applicability to component requirements.
- **Integration Test:**
  The integration test is performed to understand how effectively the component can be integrated into the selected component architecture. Components that are found suitable are candidates for inclusion in a library of reusable components.
- **Field Test:**
  To finally determine its usefulness the component should be tested in a user environment. This will show how effectively the component fulfills user and inter-operability requirements.

The described process was used for large components in the case described in [67], but can of course be used also on more granular components.

Crnkovic et al have described the development with component as being three separate but coordinated processes in [15]. These are

- **System development,**
  in which components are combined into specific products and systems based on existing or new components,
- **Component assessment,**
  which includes activities to select components that can be a part of a component repository or being selected from a repository for a specific system and product,
- **Component development,**
  which describes the activities to develop independent components and ensuring that they are made available to the intended user of the component.

On the basis of the references above and the reference models investigated, we conclude that the practices described in the models will also support component-based product development. We also see that there is a need to add specific requirements through the description of more detailed activities that can be useful, in addition to the ones currently described.

An example of how this can be done for one reference model, CMMI, is found in Table 1. The goals for CMMI related to Product Integration match the extent of the product integration process in other reference models. The additions are needed to handle the consequences of separate processes for system development, component assessment, and component development. Major parts include handling of the infrastructure for a component model, availability, and suitability of components, and interdependencies between components. The effort for performing the described activities will increase the cost for the development of systems, but the reuse of existing components in combination with higher quality of components when delivered to integration should counter and outweigh this.

2.5 Conclusion

My conclusion is that the different aspects of software engineering, such as project life-cycle models, software architecture, and the organization of the projects, must be considered and taken into account when performing product integration. For all these aspects, a careful management of the interfaces and interaction between both components in the system and the participants in the development projects is vital for success. I conclude also that the descriptions available today in different reference models are insufficiently used and additional effort is needed to make them useful.

Table 1. Proposed changes to Product Integration process in the CMMI

Specific Goal 1: Prepare for Product Integration
System development • Consider component availability when determining the integration sequence • Ensure that the chosen component model is supported by the infrastructure
Component assessment • Investigate component interdependencies
Component development • Ensure that the anticipated infrastructure is specified, i.e. component model and framework • Describe tests and expected results as a part of component specifications
SG 2: Ensure Interface Compatibility
System development • Include checks that the chosen interfaces adhere to the overall architectural decision on patterns and strategies
Component assessment • Check the consistency of interfaces
Component development • Adhering to the component model helps ensure that the definition of interfaces is complete • Check that all functions, including built-in-test facilities, can be accessed, i.e. that the defined interfaces permit the intended functionality
SG 3: Assemble Product Component and Deliver the Product
System development • (No additions proposed)
Component assessment • Prepare assemblies of components at assessment time to ensure that the components fit the system
Component development • Assemble test systems to show suitability for different applications • Test verification procedures that are a part of the component delivery • Make components available in an internal repository, or on the market

Chapter 3. Research Method

This chapter includes an overview of the research methods used in software engineering and how these are used in the research presented. Each of the papers included in the thesis contains the method applied in that part of the research as well as a discussion about validity, and limitations of the studies. The general research strategy and the overall validity are discussed here.

3.1 Method

Software engineering research uses a number of methods to ensure progress in the area. Basili has presented four approaches [68]: 1) the scientific method, 2) the engineering method, 3) the empirical method, and 4) the analytical method. The three first are classified as being part of the scientific paradigm, while the fourth is the analytical paradigm. Understanding these different methods also help distinguish research from development. Research aims at understanding a phenomenon, e.g. why and how the use of a process can help a product development organization improve performance, while development is performed to implement, e.g. to describe, and train people in the use of a process.

Software engineering research is in many respects different from other types of computer science research, and mathematics, as it heavily depends on human behavior through the people developing the software products and systems. This is described by Wohlin et al in [69] and is especially true in the research regarding processes. This makes it difficult to use the analytical paradigm.

In [70], Shaw describes different aspects of research in the area of software engineering. One of her conclusions is that initial research may result in informal and qualitative results, which give incentives for continued research. As the research in an area matures, more empirical models are presented, and finally result in formal models which justify larger

investments to introduce the research outcome on a larger scale. These different steps require different methods as the expected results differ.

One way to distinguish between different types of results from research is based on the type of research that has produced them. This has been expressed for human computer interaction by Brooks and adapted for software engineering by Shaw in [70]. It is necessary to distinguish between different types of results because research results based on experiments that are possible to control, and can show statistical results, are limited in scope, while broader results that are based on observations are more difficult to validate. It is necessary to know the background to be able to understand the implications of the presented research.

The proposed classification of research results includes Findings, Observations and Rules-of-thumb. Findings are the results from soundly-designed research, and with clear declaration of the domain for which a generalization is valid. Observations report on actual phenomena that are interesting, but may be from under-controlled environments and/or observations from limited samples. Finally Rules-of-Thumb are generalizations over a domain that is larger than the tested one. All three types of results should be judged for freshness, and it should be clear for all reports to what type the results belong. There is also a need for all three types; Observations and Rules-of-Thumb will give guidance to practitioners and help generate basis for further research that eventually could lead to Findings.

In [71], Redwine and Riddle describe different phases in software engineering research from the aspect of maturation of software technology. These are *basic research*, *concept formulation*, *development and extension*, *internal enhancement and exploration, external enhancement and exploration*, and *popularization*. Each of these phases requires different methods and tools, and will also bring the knowledge area forward in different ways. The *basic research* is used to investigate basic concepts, and to formulate basic research questions in the area. *Concept formulation* comprises the forming of a research community, and a convergence of different ideas. Solutions to specific problems are also published. The next phase, *development and extension*, includes making preliminary use of ideas and concepts and aim at a generalization of solutions and approaches. *Internal enhancement* refines the solutions and broadens the use to other domains, and the research should in this phase start to show value as it can be used to solve real problems. *External enhancement* brings the technology to other people that have not been involved in the development of the

concepts and the use of the research results shows it substantial value. Finally, the *popularization* includes a full embracement of the technology, with commercialization and an expanded user community as a result.

A substantial set of ideas and guidelines is available for product integration, but the methods are not validated. In order to understand the problems and how different activities, tools, and methods can help in achieving more efficient and effective product integration, it is necessary to have empirical data as a basis. The data in this research has been collected in an industrial environment with case studies involving project developing commercial products, which leads to an under-controlled environment. The results are therefore *observations* [70]. We have chosen to work in the *development and extension* phase as described in [71]. Our aim is thus to make use of existing ideas to determine if any generalizations can be made, and to clarify the ideas underlying what is described in the reference models.

The method used in the research presented consists of three steps. The first two steps have been made in iterations, and step three is the final analysis:

(i) Examination of existing standards and reference models that includes practices for product integration;

(ii) Based on knowledge from the reference models, case studies have been performed to obtain an understanding of the connection between the use of practices and problems found in product integration;

(iii) Analysis of the combination of the results from the case studies and the content in the reference models.

The case studies are planned and executed based on methods described by Yin [72] and Robson [73]. This includes the preparation and the implementation of the studies through interviews and document reviews, and the analysis based on the observations. The three iterations that have been performed are shown in Figure 2.

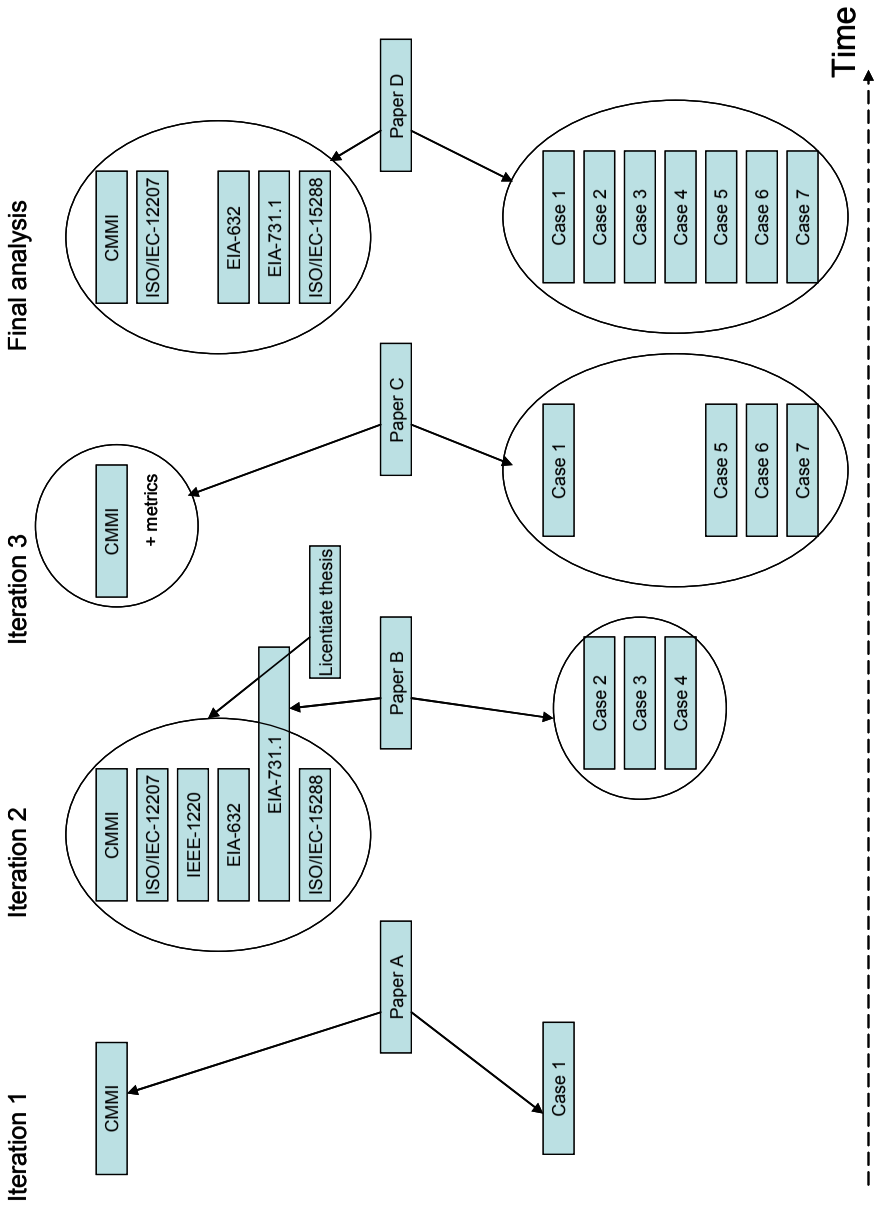

Figure 2 Research iterations

The first iteration included an investigation of CMMI [3], and a case study of one product development organization and is presented in paper A [10]. The case study included two parts; an investigation of the practices used in the company based on CMMI, and an identification of the problems found in product integration. The data for the case study was collected through interviews, document reviews, and reviews of the organizations' process documents.

The second iteration began with a study to find additional suitable reference models. A set of models containing requirements or directions for product integration were selected. The study to find suitable reference models was based on information from standardization organizations such as ISO [74], ANSI [75], and IEEE [76] and organizations such as SEI [77] and INCOSE [78]. The criterion for selecting a model was that the reference model should be relevant to product development of products that include substantial part software.

That a reference model fulfilled the criterion was determined by its purpose as in the model documentation. The result of the investigation is described in my licentiate thesis [9]. The selected reference models were:

- ISO/IEC 12207 Information technology - Software life cycle processes [2]
- EIA-632 Processes for Engineering a System [30]
- CMMI Capability Maturity Model Integration [3]
- EIA-731.1 Systems Engineering Capability Model [1]
- ISO/IEC 15288 Systems Engineering – System life cycle processes [5]
- IEEE Std 1220 for Application and Management of the Systems Engineering Process [79]

Also ISO 9001 [80] was initially considered, but was rejected since the expectations on the product integration process are limited. The standard describes the requirements on design and development, and the general requirements such as planning, input, output, review, verification, validation, and control of design and development changes are all applicable to product integration. However, as the expectations on the product integration process are not mentioned explicitly, this standard has not been analyzed further.

From among the selected reference models, EIA-731.1 [1] was chosen for use in the second case study which included three development groups in two organizations and is described in paper B [11]. The data was collected in the same way as for the first case study; through interviews, document reviews, and review of process documents.

Before the third iteration of cases was initiated, the selected set of reference models was thoroughly investigated. The analysis was performed through careful examination of the standards, and a compilation of a union of what was included in the reference models was developed.

The third and final iteration included a case study with four development organizations, and is described in paper C [12]. In addition to the data collection based on a reference model (CMMI), data was also collected from the build activities. The purpose was to see if it is possible to combine the use of a reference models with existing data from the activities in the organization. The data was collected was through interviews, document reviews, and through the collection of data made by the practitioners (e.g. build failure frequency), either automatically or manually.

Based on the three iterations of step one and two of our research method, the findings from all case studies and the investigation of reference models were analyzed as step three. The results are available in paper D [13]. The analysis was for each of the case studies performed, based on available reference models. Five reference models were selected as they had explicit expectations on product or system integration. This was an additional criterion compared to iteration 2 of the research. One reference model was excluded at this point, IEEE Std 1220 [79], as most of the references to product integration in this standard were implicit and not useful for our purpose.

All original material from the cases was used, but no additional information was collected. The problems found through the case investigations as well as the implementation of practices were mapped to all reference models. One factor in this phase was that the material for each case was collected on the basis of only one of the reference models. Through the use of the notes and reference documents collected, it was possible to determine if the practices were implemented for most cases. The only exception was for the practice related to the product strategy. Information was missing in three of the cases, and it was impossible to draw conclusions if that practice was performed or not.

There is an important distinction between insufficient support in the reference models, and the unsatisfactory implementation of good practices by the product development organizations. The difference has guided us in our research; the focus of the case studies being on insufficient use. The compilation based on all cases studies includes a discussion about the indications of insufficient support for product integration working well in the reference models.

As an additional part of the final step of our research, the relationship between architecture and the product integration process has been investigated. As architectures are developed and evolve, the implementation of the product integration process may be affected. There are several related subjects: i) interface management is an important part of software product architectures and product integration, ii) the division of subsystems and components performed in the architectural development influences the possibilities to select different integration strategies and sequences, iii) and the product integration strategy may introduce constraints on the architectural design.

This part of the research focuses on the evolution of the architecture for a system and has proposed and piloted a method to understand what changes to processes are necessary to achieve the business goals that are the reason for an architectural change. The results are presented in paper E [14]. The method was based on existing methods for assessing architectures and processes, primarily ATAM [81] and CMMI [3, 82]. The investigation in the pilot study was performed as a participant-observer study with two researchers participating in the use of the method. After the pilot study, the method was evaluated based on two criteria as defined before the study.

3.2 Validity and Limitations

Four types for validity based on Robson [73], Yin [72], and Wohlin et al [69]; construct validity, internal validity, external validity (or generalizability), and reliability (or conclusion validity) have been considered in this thesis.

The ***construct validity*** relates to the data collected and how this data represent the phenomenon investigated. This is addressed in the case studies through multiple sources for the data in the project appraisals. This is accomplished through more than one interviewee for each case as well as using document reviews. The use of reference models as a basis for the

interviews and document reviews secure that the data collected is relevant. A concern here is that we have used different reference models for the different cases. However, the collected data also includes information about other practices than those available in the reference model used for the data collection. It has been possible to use this additional data in the comparison with all reference models. If no data is available in the case material for a specific practice in any of the models this is presented in the research results. One specific problem that has been observed in the interviews is that even if an interviewee responds that a practice is performed, we have found that the activities for that practice may actually not be performed. This is treated through corroboration of data through several interviews and document reviews.

The ***internal validity*** concerns the connection between the observations and the proposed explanation for these observations. This has been addressed in several ways. For the appraisals using reference models, several steps have been taken to ensure that the mapping and understanding are correct. A detailed description of the methods used for the appraisal can be found in [16]. One risk related to the internal validity is that, through the investigations and through participation in the discussions of product integration, we affect the processes while collecting data. The results that we collect are however a combination of the performed practices and the problems occurring in the organization. Thus, the data we collect reflects the state in the organizations at the time for the data collection. The results are valid, and useful for our purposes, even if they might have been different if we had not influenced the organization through the investigation. For the case studies that have been performed in the company where I am working, there is the risk that an internal researcher would get different answers than an external. This can go both ways: persons responding to questions may be more open to an external researcher, than to an internal or vice versa. This has not been investigated specifically. One advantage in the case studies performed inside the company is that I have better background knowledge and can understand the responses and ask better clarification questions. Access to different projects has been easier through the internal case studies, and this is probably also an advantage.

The possibilities to generalize the results from a study are dealt with by studying the ***external validity.*** This is addressed through the selection of cases from different domains including telecommunication, power protection and control, process automation, and industrial robot control. The investigations cover primarily embedded systems, but workstation software

products have also been included. In these applications, the workstations are a part of a larger industrial system, typically as operator or engineering stations. The focus is on industrial applications as we see that the requirements and expectations differ from those associated with consumer products, ERP systems, and banking applications. We have also limited the research to products that are delivered to more than one customer, i.e. the cases we have investigated do not include any bespoke development. Additional aspects that have been considered to address the external validity are to ensure that the case studies include different countries and different types of organizations. One disadvantage is that several cases are from the same multinational company. However, the investigated organizations are from different divisions, have distinctly different development processes, and the products are intended for different application domains. The result is that a broad spectrum of different types of products and organizations has been investigated.

High ***reliability*** increases the possibilities to reach the same conclusions as those of another researcher repeating the study. The reliability aspect of the studies has been addressed through the detailed description of the procedure used in each case and have been included as a part of the publication for each case study. Additionally, the method for collecting data from organizations and projects use techniques described in [16].

3.3 Conclusion

In this chapter, I have described how the research presented in this thesis is based on current knowledge, theories and guidelines for software engineering research. I have also described how the planning and execution of the research as well as the selection of case studies contribute to the different aspects of validity. The conclusion of the discussions is that the validity for this research includes industrial software products, intended for use by more than one customer.

Chapter 4. Research Results

This chapter summarizes the research results and relates the research questions with the individual papers included in this thesis.

The main question for our research is to understand *what factors influence the possibilities to achieve efficient and effective product integration.* Efficient and effective product integration is manifested through a minimum delay of the flow of components to larger components or products and systems.

To investigate the factors, we have used different types of reference models. We have examined what effect the use of, or failure to use, the practices described in the reference models have on the performance in product integration. This was performed by investigating product development organizations and through examining development projects. We have further examined how changes in architecture can influence processes, and how this influence can be captured. The relationship between the areas we have investigated and the research papers A-E can be seen in Figure 3.

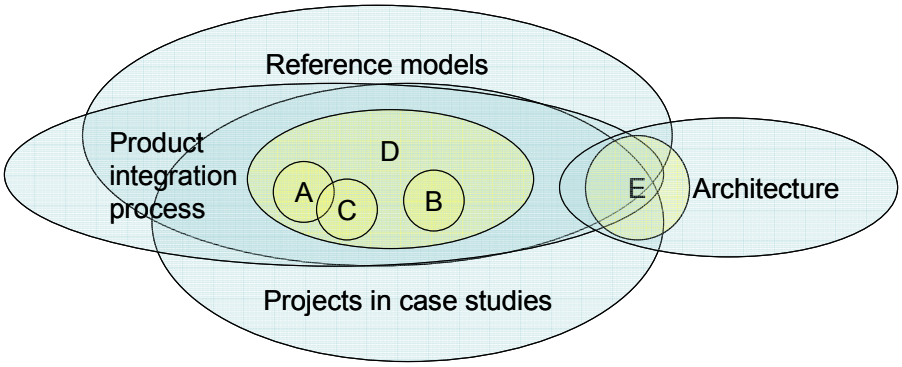

Figure 3. Relationship between the research papers A-E and the investigated areas

4.1 Results related to questions 1a and 1b

Papers A, B and C present the investigation of product integration processes in product development projects based on different reference models. Paper A and C share one of the case studies. Paper D summarizes and expands on the findings from paper A, B and C. Finally, paper E is the first step in a new research direction based on our findings intended to determine additional influences and considerations that need to be taken into account when defining and improving product development processes in general, and specifically product integration processes.

The research questions presented in section 1.2 make our research more concrete, and the response to them is based on papers included in this thesis.

The first question is used to investigate the use of current reference models:

> *Are the practices described in available reference models for product integration necessary and sufficient for visible reduction of problems in the product integration process?* **(Q1a)**

Our investigations (papers A-C) [10-12] compare the performed activities in different organizations with practices described in different reference models. The problems related to product integration have also been captured. The problems in the case studies are associated with practices, which gives us an understanding of what practices can actually help avoid product integration problems. The case studies from seven development organizations in the three papers A-C give at hand that the types of difficulties encountered in product integration can be reduced through following the practices described, but the specific practices in each of the reference models are not sufficient. In particular the different reference models cover different aspects of product integration and a parallel investigation has been directed by the question:

> *What additions and modifications are needed in the available reference models to take advantage of current body of knowledge in product integration?* **(Q1b)**

The answer to this question is a combination of an analysis of the selected reference models, and a compilation of the cases in papers A - C. The results of these steps are presented in detail in paper D [13], and are summarized here. The reference model analysis resulted in a union consisting of 15 practices which describes what can be considered the current level of knowledge in product integration.

Of the 15 practices, four are concerned with preparation of the product integration:

1. Define and document an integration strategy
2. Develop a product integration plan based on the strategy
3. Define and establish an environment for integration
4. Define criteria for delivery of components

The following five practices describe design and interface management

5. Identify constraints from the integration strategy on design
6. Define interfaces
7. Review interface descriptions for completeness
8. Ensure coordination of interface changes
9. Review adherence to defined interfaces

One practice defines the preparation of the verification to be performed in the product integration:

10. Develop and document a set of tests for each requirement of the assembled components

The actual integration of components is made up of four practices:

11. Verify completeness of components obtained for integration through checking criteria for delivery
12. Deliver/obtain components as agreed in the schedule
13. Integrate/assemble components as planned
14. Evaluate/test the assembled components

Finally, a single practice ensures that the integration is documented:

15. Record the integration information in an appropriate repository

Note that this division of practices is more detailed than is common in the reference models that have been used in this research, and may be seen as addressing the responsibility for different roles in the organization. The preparation is the responsibility of the project manager with assistance of the product integrator. The second part, interfaces, is an architectural task, involving architects and developers. The test preparation as well as the actual integration is the product integrator's responsibility, while again the project manager with the assistance of all the product integration participants will be responsible for recording the results.

As a second step to answer research question Q1b, the union of practices from the different reference models has also been compared to the different problems found in the case studies to clarify which practices will directly reduce the number and the effects of problems in product integration. A detailed description of this analysis is available in paper D [13]. The results are summarized in Figure 4 and show the product integration problems that can be related to a practice in each reference model. Our conclusion is that none of the standards include all necessary practices needed to help the organizations in avoiding the problems. As an example, for ISO/IEC 15288 we could associate 11 of the 17 problems found in the case studies to any one of the product integration practices in that standard. The result confirms the need of a broader approach than is available in any of the examined reference models. Using a combination of the examined reference models as we propose will cover activities and procedures that address all the problems encountered in our case studies. Further analysis is presented in Table 2. This analysis shows that a combination of CMMI and either ISO/IEC 15288:2002 or EIA-733.1 would be sufficient to include all needed practices.

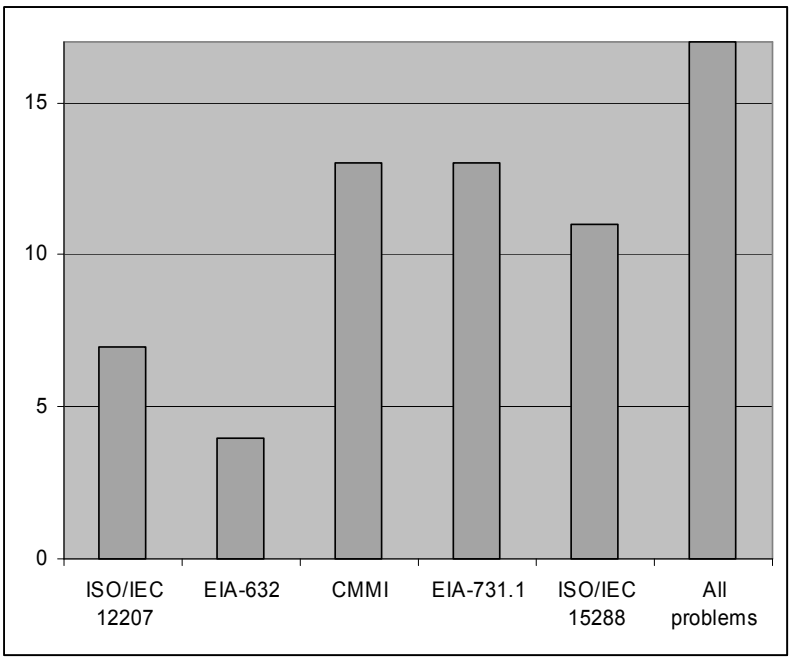

Figure 4. # of unique problems related to the practices in each standard and the total # of unique problems for all cases

Table 2. Problems from cases related to reference models

Problem identity	ISO/IEC 12207	EIA-632	CMMI	EIA-731.1	ISO/IEC 15288
Case 1, problem A	Covered			Covered	Covered
Case 1, problem B	Covered	Covered	Covered	Covered	Covered
Case 1, problem C			Covered	Covered	
Case 2, problem A			Covered		
Case 2, problem B				Covered	Covered
Case 2, problem C			Covered	Covered	
Case 3, problem A	Covered	Covered	Covered	Covered	Covered
Case 4, problem A			Covered	Covered	
Case 5, problem A	Covered			Covered	Covered
Case 5, problem B			Covered		Covered
Case 6, problem A	Covered	Covered	Covered	Covered	Covered
Case 6, problem B	Covered			Covered	Covered
Case 6, problem C			Covered		Covered
Case 6, problem D			Covered	Covered	
Case 7, problem A	Covered	Covered	Covered	Covered	Covered
Case 7, problem B				Covered	Covered
Case 7, problem C			Covered	Covered	

Of the 15 Product Integration practices, we have observed that problems are likely if any of the following five are neglected:

4, Define criteria for delivery of components

7, Review interface descriptions for completeness

8, Ensure coordination of interface changes

11, Verify completeness of components obtained for integration through checking criteria for delivery

12, Deliver/obtain components as agreed in the schedule

For PI practice 1, “Define and document an integration strategy”, and PI Practice 3, “Define and establish an environment for integration” we have seen that there may be problems even if the practices are performed.

For eight of the practices, we have not seen any problems:

2, Develop a product integration plan based on the strategy

5, Identify constraints from the integration strategy on design

6, Define interfaces

9, Review adherence to defined interfaces

10, Develop and document a set of tests for each requirement of the assembled components

13, Integrate/assemble components as planned

14, Evaluate/test the assembled components

15, Record the integration information in an appropriate repository

One important additional factor when determining what practices need to be performed is the dependencies between them. Some of the practices are necessary as a preparation for others, i.e. support the necessary practices while additional practices may be more indirectly connected.

Through reasoning about the different practices we have identified the dependencies, and the result of this analysis is presented in Figure 5.

We claim that for the interface handling, also PI practice 6 “Define interfaces” is important as PI practice 7 “Review interface descriptions for completeness” relies on it. A weaker dependency is also identified between PI practice 6 and PI practice 5 “Identify constraints from the integration strategy on design”. The same reasoning can be applied on PI practice 2 “Develop an integration plan based on the strategy” which is recognized as a prerequisite for PI 12. PI practice 11 “Verify completeness of components obtained for integration through checking criteria for delivery” depends on the checks done through PI practice 9 which is “Review adherence to defined interfaces”. Finally, PI practice 1 “Define and document an integration strategy” can be depending on PI practice 15 “Record the integration information in an appropriate repository” as the collected data is important when deciding on changes and improvements in the strategy for product integration.

A conclusion is that the set of practices that need to be followed is larger than the set that we have seen causes problems in the development organizations. The additional practices that support the crucial ones are PI

practices 2 “Develop a product integration plan based on the strategy”, 6 “Define interfaces”, 9 “Review adherence to defined interfaces”, 15 **“**Record the integration information in an appropriate repository”, and indirectly also PI practice 5 “Identify constraints from the integration strategy on design” as PI practice 6 is depending on it. Note that the dependency that PI practice 15 has on all other practices has been omitted. To be able to record the results from the PI activities so that this information can be used for future improvement, information from all activities should be included.

The remaining three practices that are not connected to or supporting the crucial practices are PI practices 10, 13, and 14. PI practice 10, “Develop and document a set of tests for each requirement of the assembled components”, is likely to give problems through later discovery of errors, and resulting problems are not connected to the product integration. Our interpretation is that this could explain why it is not connected. The same reasoning is applied to PI practice 13. “Integrate/assemble components as planned”, and PI practice 14, “Evaluate/test the assembled components”. These practices include activities that are performed if integration is performed. Failure in the practices would be that the defined sequences and procedures are not followed which would give problems in alter phases. Problems found executing PI practice 13. “Integrate/assemble components as planned”, and PI practice 14, “Evaluate/test the assembled components” are normally related to the preparation of product integration or the environment.

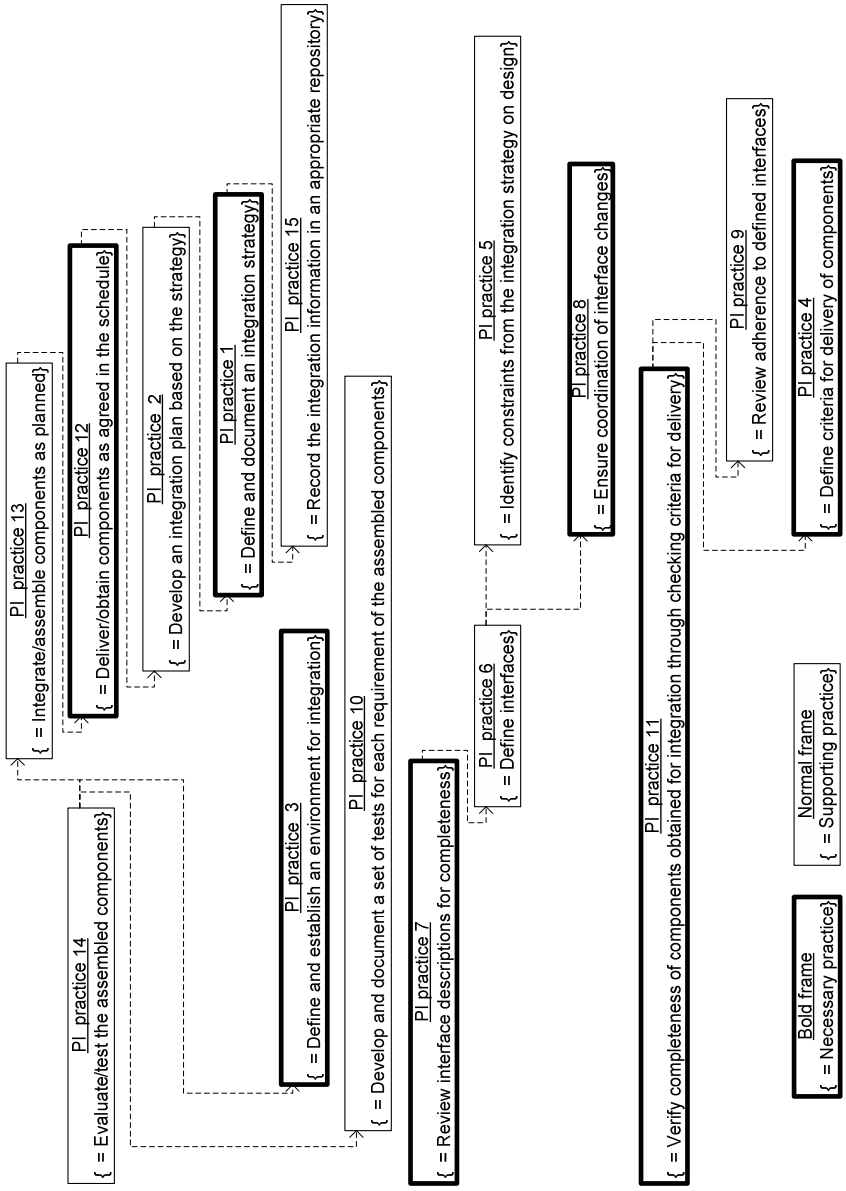

Figure 5 Dependencies between PI practices.

For PI practice 1 and 3, we need to understand how the problems occur even if our investigation indicates that the practices are followed. For PI practice 1, "Define and document an integration strategy", we have observed that problems were encountered even when this practice is followed. This may be explained by the strategy selected being les suitable for the environment of that project. PI practice 1 sets the stage for the integration process, and even if a strategy is chosen, it may not fit the specific conditions under which a project is run. However, we have not been able to determine what the root cause was, as the rationale for the selection of the strategy has not been documented. In this context, the difference between deciding on an integration strategy and the preparation of an integration plan is important.

When selecting a strategy for the product integration in a project or for an organization, several different factors will determine what will be the best technique for the sequence. Among important factors are:

- Risk elimination
- Customer needs for functionality
- Availability of components/resources(people/technology/tools)
- Technical infrastructure
- Dependencies between components (anatomy)
- Testing possibilities

McConnel describes different strategies in [4]. The first observation described is that the incremental strategies are superior to the phased (also called "big bang" integration). The techniques mentioned for the incremental integration are top-down integration, bottom-up integration, sandwich integration, risk-oriented integration, feature-oriented integration, and T-shaped integration. It is also pointed out that these techniques are to be tailored to fit each specific project, and not to be dogmatically followed – in the end, a project need to have its own integration strategy with the resulting integration sequence. One difficult issue is to measure how well this practice is followed. One proposal is to use checklists as the one presented in [4].

PI practice 3, "Define and establish an environment for integration", disclosed problems during the appraisals for two cases even when the practice was considered to be performed. A closer investigation into these cases show that the effectiveness with which the practice is performed was not considered, and can be classified as a false positive from the appraisal. It highlights however an important point: the use of practices need to be

verified by information about the performance, and an analysis of the connection between the actual performed practices and the measured results is key to improving the product integration process.

4.2 Results related to question 2

It was observed in the case studies that the architecture of a product or system is very often changed without the processes to further develop the system being altered to reflect this evolvement. One practice described in ISO/IEC 15288 is related to this and is included as PI practice 5 of the union: “Identify constraints from the integration strategy on design”.

To ensure that architectural changes are reflected in the process and strategies for product integration, tools are needed that can provide support to find the needed changes. As many organizations today start introducing changes by redefining or evolving the architecture, there is a need to understand and map the influence on process from architectural decisions. This leads to an additional research question and corresponding answers:

> *How can necessary changes in the integration process due to changes in the product architecture be identified and implemented?* **(Q2)**

Through an investigation of different models used for supporting architectural decisions, and appraisal methods for process improvement, a method has been proposed and piloted (paper E [14]). The method was successful in helping the organization to understand what process changes are needed to benefit from the architectural changes. This was especially true for the product integration process as the architectural changes called for new strategies. This fact was not identified by all parts of the organization before the use of the proposed method. The proposed method has also been updated based on the results from the study. An additional result from the investigation is that the understanding for needed process changes, including steps in product integration, has increased in the pilot organization.

4.3 Conclusion

From our case studies I conclude that the available knowledge regarding the product integration process is inconsistently used by different product development organizations.

It can be questioned if the use of the PI Practices is valid for all types of projects and organizations. As seen in the analysis above, all the practices described are either crucial in them selves, supporting the crucial ones, giving problems later in the process, or dependent on other practices and thus not indicating any problems. My conclusion is that all practices are needed, but that an adaptation is necessary to get the proper level of activities in each project. A future research area can be to investigate the need for a method for the adaptation.

When investigating the product integration area, we have seen that organizations are aware of practices that are described in reference models. However, as the information in the models is too limited, the usefulness is also limited and additional information such as examples and hands-on methods are needed. Consequently, the models should primarily be used as guidelines for what to improve, and information about how the practices should be implemented need to be found elsewhere.

I also conclude the product integration processes may be influenced by evolvement of the architecture and design of software systems, and that the method provided, the needed changes of processes can be understood and implemented.

Chapter 5. Conclusions and Future Work

The goal of the research presented in this thesis has been to understand what factors are most important when trying to achieve efficient and effective product integration, and how these factors influence the software product development processes.

The origin of the research is in the needs identified in organizations developing products for industrial use with significant software content. These needs have been confirmed in case studies. Organizations experience problems in product integration due to insufficient and inconsistent strategies and plans for integration, lack of understanding of how interface management and other architectural decisions influence product integration, and inadequate control of components delivered for integration. The focus in the research is on industrial software products, with real-time requirements. This implies specific needs to understand and be able to manage quality attributes, such as performance, reliability and availability of the resulting product.

I have through investigations of information available in reference models regarding product integration practices and a series of case studies identified the key elements for software product integration practices. These have been organized in five categories: preparation of the product integration, design and interface management, preparation of the verification to be performed, the execution of product integration, and documentation of the product integration results. The collection includes the practices available today in relevant reference models, which have been made accessible through the compilation. The validity of the practices has been examined through case studies. The work to reach an agreed body-of-knowledge for software product integration processes should be continued. This can be done through relevant research in other application domains, and reference models applicable for these domains.

Table 3. Collection of Product Integration Practices

Preparation of product integration	
Define and document an integration strategy	Necessary
Develop a product integration plan based on the strategy	Supporting
Define and establish an environment for integration	Necessary
Define criteria for delivery of components	Necessary
Design and interface management for product integration	
Identify constraints from the integration strategy on design	Supporting
Define interfaces	Supporting
Review interface descriptions for completeness	Necessary
Ensure coordination of interface changes	Necessary
Review adherence to defined interfaces	Supporting
Preparation of product integration verification	
Develop and document a set of tests for each requirement of the assembled components	Problems likely related to other practices or processes
Integration of components	
Verify completeness of components obtained for integration through checking criteria for delivery	Necessary
Deliver/obtain components as agreed in the schedule	Necessary
Integrate/assemble components as planned	Problems likely related to other practices or processes
Evaluate/test the assembled components	Problems likely related to other practices or processes
Documentation of the integration	
Record the integration information in an appropriate repository	Supporting

The collection of practices that I have described provides support for software product development organizations. The collection is summarized in Table 3. Of the 15 practices, there are indications in our case studies that five are necessary, as shown in the rightmost column in Table 3, to avoid problems in product integration. Two practices have been seen to be necessary, but that problems arise if the practices are inadequately performed. An additional five practices support the practices considered necessary. For the remaining three practices, there are no indications that organizations will have problems if not implementing them. However, the nature of these three practices is such that any problems would most likely be related to other practices such as the preparation of the integration or integration environment, or verification.

The influence that architectural decisions have on product development processes is seldom investigated in the industry. Through a case study, we have demonstrated the usefulness of a method to examine and identify changes necessary to be made to development processes. When evolving the architecture of the product in a case study, the processes influenced include the product integration process. By providing a method to understand how different changes affect the processes, proposed improvements for better product integration can be understood and assessed.

The research presented here has been performed based on available theories and guidelines for research in software engineering, and care has been taken to address different types of validity. This is done through careful planning and execution of the studies, and through selecting relevant case study organizations working with software products in industrial applications. The conclusion is that the validity for this research includes industrial software products, intended for use by more than one customer.

Future research includes additional validation of the collection of practices, also in other application domains. A subject which needs to be investigated is implementation of proposed practices, to understand why available practices are not used, and why the implementation sometimes fails.

The impact the presented research will have when applied in industry remains to be seen and additional investigations are needed to explore this. Each organization using the practices described in this thesis needs to implement the practices, adapted to the organizations needs. Further investigations are needed to understand how an impact can be achieved with reasonable effort.

Additional research is also needed to look at other methods, tools, and technologies to help product development organizations improve product integration. Through the compilation of practices based on the available reference models and an understanding how these can help, a foundation is available for future research. This can also be the starting point to investigate different types of project and development models to understand if there are specific requirements that should be taken into account.

References

[1] EIA-731.1, "Systems Engineering Capability Model," Electronic Industries Alliance, 2002.
[2] ISO/IEC12207:1995, "Information technology - Software life cycle processes," ISO/IEC, 1995.
[3] SEI, "CMMI® for Development, Version 1.2.," Pittsburgh, PA, USA, Technical Report CMU/SEI-2006-TR-008, 2006.
[4] S. McConnel, Code Complete, 2nd ed. Redmond, Wa, USA: Microsoft Press, 2004.
[5] ISO/IEC15288:2002, "Systems engineering - Systems life cycle processes," ISO/IEC, 2002.
[6] J. Campanella, Principles of Quality Costs: Principles, implementation and Use, 3rd ed. Milwaukee, WN, USA,: ASQ Press, 1999.
[7] RTI, "The Economic Impacts of Inadequate Infrastructure for Software Testing." Gaithersburg, MD, USA,: National Institute of Standards and Technology, 2002.
[8] M. Bajec, D. Vavpoti, and M. Krisper, "Practice-driven approach for creating project-specific software development methods," Information and Software Technology, vol. 49, pp. 345, 2007.
[9] S. Larsson, "Improving software product integration." Västerås: Dept. of Computer Science and Electronics Mälardalen University, 2005, pp. xi, 108.
[10] S. Larsson, I. Crnkovic, and F. Ekdahl, "On the expected synergies between component-based software engineering and best practices in product integration," presented at Proceedings - 30th EUROMICRO Conference, Aug 31-Sep 3 2004, Rennes, France, 2004.
[11] S. Larsson and I. Crnkovic, "Case Study: Software Product Integration Practices," presented at 6th international conference Profes, June, 2005, Oulu Finland, 2005.
[12] S. Larsson, P. Myllyperkiö, and F. Ekdahl, "Product Integration Improvement Based on Analysis of Build Statistics," presented at ESEC/FSE, Dubrovnik, Croatia, 2007.

[13] S. Larsson, P. Myllyperkiö, F. Ekdahl, and I. Crnkovic, "Examination of Product Integration Practices in Reference Models," Submitted to Information & Software Technology, 2007.
[14] S. Larsson, A. Wall, and P. Wallin, "Assessing the Influence on Processes when Evolving the Software Architecture," presented at IWPSE 2007, Dubrovnik, Croatia, 2007.
[15] I. Crnkovic, M. Chaudron, and S. Larsson, "Component-Based Development Process and Component Lifecycle," presented at Software Engineering Advances, International Conference on, 2006.
[16] F. Ekdahl and S. Larsson, "Experience Report: Using Internal CMMI Appraisals to Institutionalize Software Development Performance Improvement," presented at 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 06), 2006.
[17] S. Larsson and F. Ekdahl, "Selecting CMMI Appraisal Classes Based on Maturity and Openness," presented at Product Focused Software Process Improvement (PROFES), Kansai Science City, Japan, 2004.
[18] J. O. Grady, System Integration: CRC press, 1994.
[19] G. R. Djavanshir and R. Khorramshahgol, "Key Process Areas in Systems Integration," in IT Professional, vol. 9, 2007, pp. 24-27.
[20] A. P. Sage, Charles L. Lynch, "Systems integration and architecting: An overview of principles, practices, and perspectives," Systems Engineering, vol. 1, pp. 176-227, 1998.
[21] R. Land, "Software Systems In-House integration," in Department of Computer Science and Electronics: Mälardalen University, 2006.
[22] E. G. Nilsson, E. K. Nordhagen, and G. Oftedal, "Aspects of systems integration," presented at Systems Integration, 1990. Systems Integration '90., Proceedings of the First International Conference on, 1990.
[23] D. Garlan, "Software architecture: a roadmap," presented at Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000.
[24] I. Gorton, Essential Software Architecture: Springer, 2006.
[25] F. A. Cummins, Enterprise Integration: An Architecture for Enterprise Application and Systems Integration: John Wiley & Sons, 2002.
[26] D. S. Linthicum, Enterprise Application Integration: Addison-Wesley, 1999.
[27] W. A. Ruh, F. X. Maginnis, and W. J. Brown, Enterprise Application Integration: Addison-Wesley, 2000.

[28] H. R. Parsaei, J. M. Usher, and U. Roy, Integrated Product and Process Development: Methods, Tools, and Technologies: Wiley-IEEE, 1998.
[29] C. V. Ramamoorthy, "Distributed techniques in software system integration," presented at Workshop on Future Trends of Distributed Computing Systems, Cheju Island, Korea, 1995.
[30] ANSI/EIA-632-1999, "Processes for Engineering a System." Government Electronic and Information Technology Association: Electronic Industries Alliance, 1999.
[31] D. L. Parnas, "Information distribution aspects of design methodology," presented at Information Processing 71 Proceedings of the IFIP Congress 1971 Volume 1, 23-28 Aug. 1971, Ljubljana, Yugoslavia, 1972.
[32] V. Stavridou, "Integration standards for critical software intensive systems," presented at Software Engineering Standards Symposium and Forum, 1997. 'Emerging International Standards'. ISESS 97., Third IEEE International, 1997.
[33] B. Fitzgerald, "An empirical investigation into the adoption of systems development methodologies," Information & Management, vol. 34, pp. 317-328, 1998.
[34] S. McConnel, Rapid development. Redmon, Wa, USA: Microsoft press, 1996.
[35] R. S. Pressman, Software Engineering, 6th ed: McGraw-Hill, 2005.
[36] B. W. Boehm, "A spiral model of software development and enhancement," Computer, vol. 21, pp. 61-72, 1988.
[37] B. W. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, "Using the WinWin spiral model: a case study," Computer, vol. 31, pp. 33-44, 1998.
[38] M. Fowler, "Continuous Integration, http://www.martinfowler.com/articles/continuousIntegration.html," 2006.
[39] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice: Addison-Wesley Professional, 2003.
[40] P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a coordination tool in multi-site software development," Software Process: Improvement and Practice, vol. 8, pp. 233-247, 2003.
[41] S. D. Eppinger, "A planning method for integration of large-scale engineering systems," presented at International Conference on Engineering Design, ICED, Tampere, Finland, 1997.

[42] A. Mehta and G. T. Heineman, "Evolving legacy system features into fine-grained components," presented at Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, 2002.
[43] F. A. Cioch, J. M. Brabbs, and L. Sieh, "The impact of software architecture reuse on development processes and standards," The Journal of Systems and Software, vol. 50, pp. 221-236, 2000.
[44] M. Schulte, "Model-based integration of reusable component-based avionics systems - a case study," pp. 62-71, 2005.
[45] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-integrated development of embedded software," Proceedings of the IEEE, vol. 91, pp. 145-164, 2003.
[46] E. A. Lee, "Embedded Software," in Advances in Computers, vol. 56, M. Zelkowitz, Ed.: Elsevier, 2002.
[47] J. Sztipanovits and G. Karsai, "Embedded Software: Challenges and Opportunities," in Embedded Software: First International Workshop, EMSOFT 2001, Tahoe City, CA, USA, October, 8-10, 2001, Proceedings, 2001, pp. 403.
[48] M. Svahnberg, J. V. Gurp, and J. Bosch, "On the Notion of Variability in Software Product Lines," presented at Working IEEE/IFIP Conference on Software Architecture (WICSA'01), Amsterdam, The Netherlands, 2001.
[49] P. Clements and L. Northrop, Software product lines: practices and patterns, 2002.
[50] K. Pohl, G. Böckle, and F. J. v. d. Linden, Software Product Line Engineering: Foundations, Principles and Techniques: Springer, 2005.
[51] SEI, "Software Product Lines," http://www.sei.cmu.edu/productlines/index.html, 2007.
[52] SEI, "A Framework for Software Product Line Practice, Version 5.0," in Software System Integration. http://www.sei.cmu/productlines, 2007.
[53] C. W. Krueger, "New methods in software product line practice," Commun. ACM, vol. 49, pp. 37-40, 2006.
[54] J. D. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," IEEE Transactions on Software Engineering, vol. 29, pp. 481, 2003.
[55] D. J. Paulish, M. Bass, and J. D. Herbsleb, "Global software development at siemens: experience from nine projects," presented at Software Engineering, 2005. ICSE '05. Proceedings of the 27th International Conference on, 2005.

[56] C. Van den Bulte and R. K. Moenaert, "The effects of R&D team co-location on communication patterns among R&D, marketing, and manufacturing," Management Science, vol. 44, pp. S1-S18, 1998.
[57] M. E. Sosa, S. D. Eppinger, and C. M. Rowles, "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," Management Science, vol. 50, pp. 1674-1689, 2004.
[58] S. Komi-Sirvio and M. Tihinen, "Lessons learned by participants of distributed software development," Knowledge and Process Management, vol. 12, pp. 108-122, 2005.
[59] I. Crnkovic, "Component-based software engineering - new challenges in software development," Software Focus, vol. 2, pp. 127-133, 2001.
[60] G. T. Heineman and W. T. Councill, Component-based Software Engineering, Putting the Pieces Together: Prentice-Hall, 2001.
[61] A. H. Dogru and M. M. Tanik, "A process model for component-oriented software engineering," IEEE Software, vol. 20, pp. 34-41, 2003.
[62] R. van Ommering, "Building product populations with software components," presented at Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, 2002.
[63] I. Crnkovic, S. Larsson, and M. Chaudron, "Component-based development process and component lifecycle," presented at 27th International Conference on Information Technology Interfaces, 2005., 2005.
[64] M. Morisio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft, and S. E. Condon, "COTS-based software development: Processes and open issues," Journal of Systems and Software, vol. 61, pp. 189-199, 2002.
[65] V. Tran, L. Dar-Biau, and B. Hummel, "Component-based systems development: challenges and lessons learned," presented at Eighth IEEE International Workshop on Incorporating Computer Aided Software Engineering, 1997.
[66] M. de Jonge, "Package-based software development," presented at Euromicro Conference, 2003. Proceedings. 29th, 2003.
[67] R. W. Lichota, R. L. Vesprini, and B. Swanson, "PRISM Product Examination Process for component based development," presented at Assessment of Software Tools and Technologies, 1997., Proceedings Fifth International Symposium on, 1997.

[68] V. R. Basili, "The Experimental Paradigm in Software Engineering," 2000.
[69] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in software engineering; an introduction. Norwell, Massachusetts, U.S.A.: Kluwer Academic Publishers, 1999.
[70] M. Shaw, "What makes good research in software engineering?," STTT - International Journal on Software Tools for Technology Transfer, vol. 4, pp. 1-7, 2002.
[71] S. T. Redwine and W. E. Riddle, "Software technology maturation," in Proceedings of the 8th international conference on Software engineering. London, England: IEEE Computer Society Press, 1985, pp. 189-200.
[72] R. K. Yin, Case Study Research: Design and Methods, 3rd ed: Sage Publications, 2003.
[73] C. Robson, Real World Research, 2nd ed: Blackwell Publishers, 2002.
[74] ISO, "International Standardization Organization," http://www.iso.org, 2007.
[75] ANSI, "American National Standards Institute," http://www.ansi.org/, 2007.
[76] IEEE, "The Institute of Electrical and Electronics Engineers," http://www.ieee.org/, 2007.
[77] SEI, "Software Engineering Institute," http://www.sei.cmu.edu/, 2007.
[78] INCOSE, "International Counsil on Systems Engineering," http://www.incose.org/, 2007.
[79] IEEE1220-2005, "IEEE Standard for Application and Management of the Systems Engineering Process," Institute of Electrical and Electronics Engineers, 2005.
[80] ISO9001:2000, "Quality management systems -- Requirements," ISO, 2000.
[81] R. Kazman, M. Klein, and P. Clements., "ATAM: Method for architecture evaluation. CMU/SEI-2000-TR-004," Carnegie Mellon University, Software Engineering Institute, Technical Report CMU/SEI-2000-TR-004, 2000.
[82] SEI, "Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2: Method Definition Document," Carnegie Mellon University, Software Engineering Institute, Handbook CMU/SEI-2006-HB-002, 2006.