Mälardalen University Press Licentiate Theses
No. 51

IMPROVING SOFTWARE PRODUCT INTEGRATION

Stig Larsson

2005

Department of Computer Science and Engineering
Mälardalen University

ISSN 1651-9256
ISBN 91-88834-65-4
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

ABSTRACT

The idea with product integration is that separate components are combined into a working system. However, this process of assembling parts into bigger units, products and systems is not well performed in industry, especially not when a substantial part of the product functionality is implemented in software. Many faults that are introduced in early phases are found as late as in the product integration phase, or even worse, in the verification or validation of the final delivery, or after delivery of the product or system. This leads to high costs for error correction and additional efforts for re-testing. There is consequently a need to further investigate the area of product integration to understand how the performance can be improved. Different practices have been described in standards and models, but the area is still under development. No widely agreed upon body-of-knowledge has so far been defined for product integration.

A large part of the development of products containing software for industrial use is conducted in small or medium sized teams. This requires that any data collection methods used to acquire reliable information regarding performance in a project or organization minimize the intrusion. A facilitating approach was needed to understand how units with distinct characteristics should be approached. Based on several years of interaction with different types of organization, the presented research includes an analysis of various methods for data collection. The result is a proposed method for selecting different sizes of investigations based on the openness and maturity of the organization.

The main purpose of this research is to understand which factors influence the integration process and what can be done to improve the execution of it. It includes investigations to understand if the described best practices are appropriate, and if there are other means to achieve successful product integration. The research combines investigations of existing compilations of best practices with case studies in industry.

Our conclusion is that the type of organization that we have investigated can reduce problems in the product integration process by following the basic practices described in standards and reference models. Problems found in product integration can in most cases be related to the fact that the organization does not follow the proposed practices. The investigations have revealed that the practices are not used in a sufficient way, that additional efforts must be put into fulfilling the requirements in standards and models, and that it is difficult to implement the practices. We have also found indications that specific technology, component based software, may assist in executing the practices. Finally, we conclude that not all standards and models include support to avoid all types of problems in product integration. This is an indication that the on-going development of the area is necessary and that an increased agreement on what can be considered to be best practices is needed.

ACKNOWLEDGMENTS

I would first like to thank my supervisor professor Ivica Crnkovic for all the enthusiasm, guidance and help throughout the work with the present thesis. From the very beginning, I have received very valuable support and encouragement.

Thanks goes also to my assistant supervisors Fredrik Ekdahl, for always being there to challenge and discuss my ideas, and Johan Schubert, for support and guidance in how to understand what research is and how it differs from work in industry. I am truly grateful for all discussions and advice.

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

Thanks to all my present and past colleagues at ABB and at Mälardalen University, especially Magnus Larsson and Rikard Land for inspiration and feedback, and to everyone taking part in the case studies that form the basis for this thesis.

Thanks to all my friends and relatives for being patient with me when time is too short.

Special thanks to my sister and mother for all support over the years, and to my father, who passed away too early, for inspiration.

Finally, I would like to express all my love and gratefulness to my wife AnnKi and my daughter Camilla.

Stig Larsson
Västerås, May, 2005

LIST OF PUBLICATIONS

The following peer-reviewed papers have been published at international conferences and are included in this thesis;

A. ***Are limited Non-intrusive CMMI-based Appraisals Enough?***
In Proceedings of the ESEIW 2003 Workshop on Empirical Studies in Software Engineering WSESE 2003, Rome, Italy, September 2003
Authors: **Stig Larsson**, Fredrik Ekdahl

B. ***Selecting CMMI Appraisal Classes Based on Maturity and Openness***
In PROFES 2004 Conference, Kansai Science City, Japan, April 2004
Authors: **Stig Larsson**, Fredrik Ekdahl

C. ***On the Expected Synergies between Component-Based Software Engineering and Best Practices in Product Integration***
In Euromicro Conference, Rennes, France, August 2004
Authors: **Stig Larsson**, Ivica Crnkovic, Fredrik Ekdahl

D. ***Case Study: Software Product Integration Practices***
Accepted to PROFES 2005 Conference, Oulu, Finland, June 2005
Authors: Stig Larsson, Ivica Crnkovic

E. ***Expected Influence of Ethics on Product Development Processes***
Technical report, accepted in a shorter version to ECAP-2005 Conference, Västerås, Sweden, June 2005
Author: **Stig Larsson**

The author has also co-authored the following peer-reviewed articles and papers that have been published at international conferences and in journals;

F. ***Workshop on Component-based Software Engineering: Composing Systems from Components***
In ECBS 2002 Proceedings, Lund, Sweden, 2002, IEEE, ACM
Authors: Ivica Crnkovic, Stig Larsson, Judith Stafford

G. ***Combining Models for Business Decisions and Software Development***
In Euromicro Conference, Dortmund, Germany, September 2002
Authors: Christina Wallin, Fredrik Ekdahl, Stig Larsson

H. ***Integrating Business and Software Development Models***
IEEE Software, 19(6):28-33, November 2002, IEEE Computer Society
Authors: Christina Wallin, Fredrik Ekdahl, Stig Larsson

I. ***Towards and efficient and Effective Process for Integration of Components-Based Software Systems***
In SERPS'03, Proceedings of the 3rd Conference on Software Engineering Research and Practice in Sweden, Lund, Sweden, 2003
Author: Stig Larsson

J. ***Concretizing the Vision of a Future Integrated System – Experiences from Industry***
Accepted to ITI 2005 Conference, Dubrovnik, Croatia, June 2005
Authors: Rikard Land, Ivica Crnkovic, Stig Larsson

K. ***Component-based Development Process and Component Lifecycle***
Accepted to ITI 2005 Conference, Dubrovnik, Croatia, June 2005
Authors: Ivica Crnkovic, Michel Chaudron, Stig Larsson

L. ***Processes Used during Software Integration – Experiences from Industry***
Accepted to Euromicro Conference, Porto, Portugal, September, 2005
Authors: Rikard Land, Ivica Crnkovic, Stig Larsson

The author has also co-authored the following presentations;

M. ***Using selected level 3 process areas to maintain level 2 focus***
Presented at European SEPG, London, June, 2004
Authors: Stig Larsson, Fredrik Ekdahl

N. ***Driving improvement using CMMI Class B and C appraisals - some experiences and lessons learnt***
To be presented at European SEPG, London, June, 2005
Authors: Fredrik Ekdahl, Stig Larsson

TABLE OF CONTENTS

1 BACKGROUND AND MOTIVATION

"Oh, I'm on my way I know I am, but times there were when I thought not"

Cat Stevens

In the Trial Version 1.0 of the "Guide to the Software Engineering Body of Knowledge, SWEBOK" [1] from May 2001, integration of components was put in appendix D. The committee was in disagreement about the existence of a generally accepted body of knowledge on the topic of Component Integration. In the 2004 version [2], the topic has been integrated as a part of the "Software Construction" chapter, where the whole text about integration reads

> "A key activity during construction is the integration of separately constructed routines, classes, components, and subsystems. In addition, a particular software system may need to be integrated with other software or hardware systems.
>
> Concerns related to construction integration include planning the sequence in which components will be integrated, creating scaffolding to support interim versions of the software, determining the degree of testing and quality work performed on components before they are integrated and determining points in the project at which interim versions of the software are tested."

The recommended reading is limited to three references. One conclusion may be that the area of product integration is rather new, and that the research community has paid little attention to the area. The reason for this may be that the area has been included in development or verification research. One example where this is the case is in the area of component-based software engineering [3-5]. On the other hand, descriptions of different strategies for integration of software can be found in textbooks such as "The Art of Software Testing" by G.J. Myers from 1979 [6], so there should be extensive knowledge about integration among software developers. The product integration area is no doubt only marginally examined as a research topic in itself.

Throughout my career as a software developer, project manager and line manager as well as a process advisor, I have seen a great deal of problems in product integration; builds crash, tests are delayed, performance of systems turns out to be a fraction of the anticipated, and stakeholders are annoyed at each other. The focus among the engineers and testers is on correcting errors and investigating problems rather than confirming the proper operation of the product or system. The relevance of these observations is confirmed by different investigations [7,8] and is a basis for this research.

Product Integration does mean different things to different readers. In this thesis, Product Integration represents the process that is performed when parts are combined into more complex parts and finally to complete products. Critical elements in product integration include descriptions and management of interfaces, the sequence in which components are integrated, and communication between different stakeholders [9]. There is also a question of overall system requirements and properties. These cannot be specified and tested on a component level, but must be handled on system level. Even if all interfaces are correct such properties may not be met.

When investigating the efficiency and effectiveness of product development both process, product, technology, environment and people aspects need to be considered [10]. All factors influence each other and in an industrial environment, it is difficult to single out any one of them as being more important than the others. Also, in the effort of improving one area of product development the solution may come from a combination of any of these aspects. The research in this thesis is focused on the process aspect, but also the technical and people aspects are taken into account, while the product and environment has been selected as boundary conditions.

One important observation was made early in the work leading up to this thesis. When integration is discussed with engineers, the association they make is very often the integration performed when deciding on the technical solution for a product [11]. This architectural integration takes place when the design of the overall system is made. This aspect of integration is not covered in this thesis.

During the work that forms the basis for a thesis, the decision on scope is probably the hardest. To cover enough to make the work interesting from a generic and theoretical perspective at the same time as being focused and concrete enough is a challenge as the direction changes from time to time. Help comes from the education in the graduate program. After many years in industry, it is a challenge to start to look at the world in new ways, with new tools such as research methods and strategies. It is rewarding, however, as the understanding of many observed phenomena increases as the perspective changes.

Product integration is the process that enables an organization or a project to finally 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 occasion where the final result can be observed. Consequently, the integration phase represents a highly critical part of the product development process. Descriptions of good practices for product integration are available [9,12-15], but my own experience as well as several other research results [7,8] show that these are not always used. The reason may be that they sometimes are not fully understood or that they are perceived as not applicable. The inability to follow good practices leads to problems as the result of the integration process is a product that accumulates all positive and negative contributions from earlier phases. Product integration is the last phase

where new functionality is created. In addition, the challenge is enhanced as requirements and designs give a more abstract definition of a software product than for a hardware product. In summary, there is a need to further investigate the area of product integration to understand if the practices described in models and standards are appropriate, and to examine if other means to reach the goal of successful product integration are available.

The product integration process is directly related to many other activities executed in product development. Figure 1 shows some of the related process areas as described by the CMMI [9].

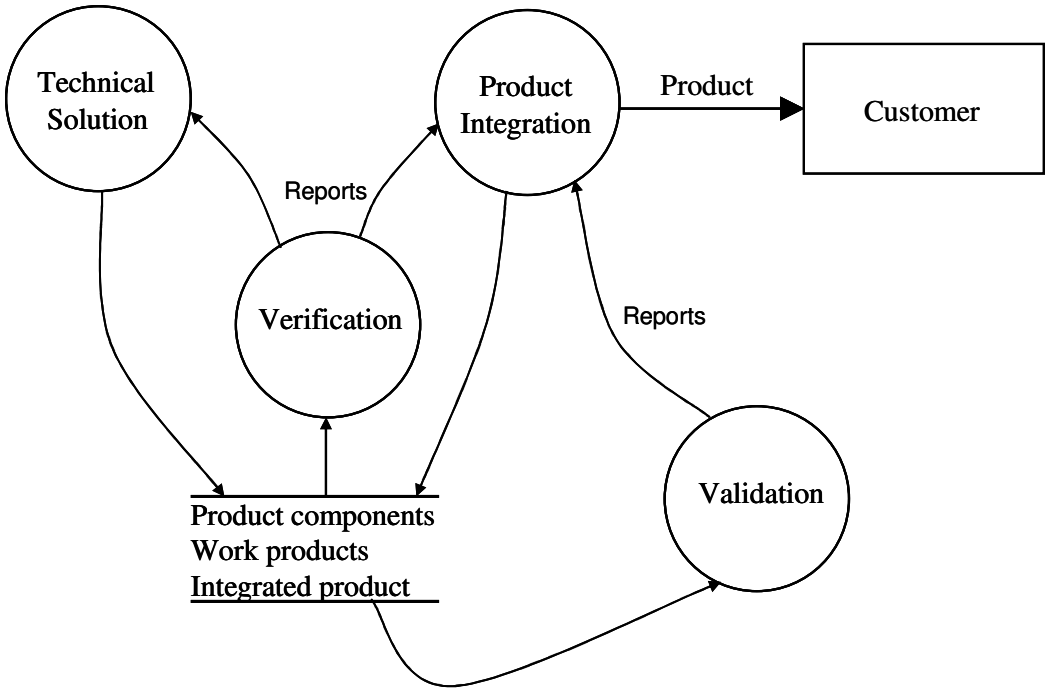

Figure 1. Processes related to product integration (based on [9]).

The activities performed in the technical solution processes result in parts or product components. Also, the necessary information about each part that is used in the product integration process is developed in parallel to the product component. Both the technical solution and the product integration processes rely on the verification processes to ensure that product components meet specified requirements. The validation process is intended to confirm that the expectations of the customer are fulfilled. Software product integration is typically performed according to a defined plan, either in one stage or incrementally. As the process of integrating products involves many different engineering disciplines such as development, architectures and testing, communication between stakeholders is vital. Successful communication depends on the ability to define and adhere to definitions and rules regarding the

concerned interactions. Standards and other reference models have described different practices [12-15], but in reality, friction between engineers performing the tasks of development, integration, verification and validation have been observed [7,8]. This indicates that additional knowledge about product integration and support for the interactions between different stakeholders are needed to achieve efficiency and effectiveness.

Integration is also easier to achieve if the parts that are to be integrated are well defined. If clear and precise technical definitions are missing, this leads to more complicated inspections at integration time, requiring more knowledge and information about each component. This includes safeguarding that the right interfaces are used and that the environment is suitable for the component. The lack of well defined interfaces and components also makes it harder to automate the different checks and tests.

By improving the product integration process, there are great advantages both for process steps that precedes the integration and the steps that follow. Clearer requirements and expectations will allow accurate deliveries to integration while a well working integration process will increase the probability for high quality products and timely deliveries to verification and validation activities.

The compilation of a thesis is in a way a type of product integration. The different conference contributions and articles need to fit together and become one product. New parts that tie the existing pieces together and put them into perspective should be created. Enough information must be included to help the readers understand the context and, if necessary, be able to find background information and further details. On the other hand, the amount of information should be limited to the necessary, leaving out speculations and irrelevant information. If this integration is successful, it should provide the reader with a tool to achieve a deeper understanding of different aspects that affect product integration. It should also be possible to use it as a basis for improving an organization's product integration process or as a starting point for further research.

The structure of this thesis is as follows. The first part of the thesis, chapter 1 through 5, contains the background and results of the research. Chapter 1 provides a background and motivation of the research. Chapter 2 includes an overview of different strategies for research in software engineering, the research strategy used for this thesis and a description of the research methods used. Chapter 3 summarizes the included papers and describes the contribution. It also contains a discussion on the validity of the research results. Chapter 4 provides an overview of state-of-the-art for product integration as well as a comparison between standards, models and the case studies performed. Chapter 5 formulates the conclusions as well as directions for future work. The second part of the thesis consists of the included five research papers.

2 RESEARCH STRATEGY

"*Insight, untested and unsupported, is an insufficient guarantee of truth.*"

Bertrand Russell

This chapter contains an overview of the challenges in Software Engineering research as well as research strategies and methods used in the study of software engineering with focus on case studies related to our research. It also contains a description of the strategy for the research described in this thesis. Finally the research questions and the approach taken to investigate each of them are described.

2.1 Research strategies in Software Engineering

Research in software engineering has matured over a number of years, but still no clear guidelines are available and the discussion is progressing in different conferences and workshops, such as [16]. The needs for a classification and characterization have been described in various publications [17-19].

Software Engineering is an engineering discipline and research in the area is primarily directed towards study of tools and methods for producing quality software products and solutions to encountered problems. This has been expressed as finding practical solutions in the real world to practical problems found in the real world [20]. As the real world is difficult to investigate, we sometimes need to investigate the problem in a research setting, and find solutions to this idealized problem. Once a suggested solution exists, two validation tasks emerge as illustrated in figure 2. The first task is to validate that the solution solves the idealized problem in the research setting. If this is the case, there is also a need to validate the solution in a real world environment and ensure that the solution also is applicable to the real world practical problem.

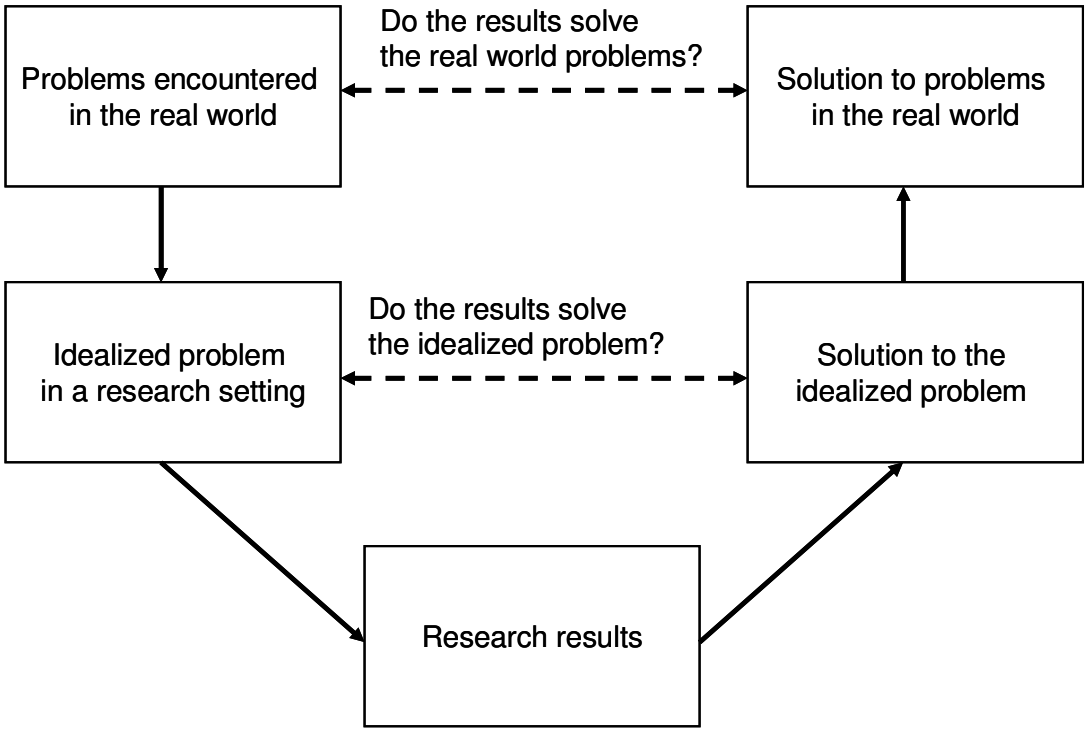

Figure 2. Development of research results (based on [20])

The use of empirical studies in software engineering research has been discussed since the mid-eighties [21], and different research strategies as well as ways to classify them have been the subject of study [22-24]. For studies of software engineering in an *industrial setting*, methods similar to the ones used in research in social sciences can be utilized. The reason for this is the similar context; the large number of factors influencing the studied systems which lead to difficulties to have full control, the fact that human beings are important parts of the processes one tries to investigate, and that the studied processes are complex. In [25], five different strategies for conducting research in social sciences are listed: experiments, surveys, archival analysis, history and case studies. Of these, the experiments, surveys and case studies are mentioned in [22] and [26] as strategies suitable for software engineering research. Experiments are done in a controlled environment, a laboratory, where a single or a few changes are inserted by the researcher. The need to control the environment limits the size of the investigation. Surveys are used to collect data from a large number of study objects, normally through sampling. This makes it possible to collect data that can show correlation between different factors without introducing changes into the environment. Case studies are observational and do not separate the object studied from its environment. This complicates the possibilities to relate the observations to the problem that is investigated.

When selecting a research strategy among these three, the main things to consider are what type of research question is investigated and how much control is required or can be obtained over the investigated phenomenon [25]. Experiments require a high degree of control which enables the researcher to insert a variation in only one or a few variables in the environment, often to make it possible to determine the effects of a change with statistical significance. The types of questions that can be answered with experiments are "how?" and "why?" as thorough knowledge about the correlation between the manipulated variables and the results are likely to be the result.

Surveys are used to sample a population to give qualitative or quantitative data to investigate a research question and are normally looking at a large number of teams, projects or organizations. Surveys are used to give response to research questions such as "who", "what", "where", "how many" or "how much".

Case studies are in general used to look at a single or a limited number of projects or organizations, which are considered as typical, to respond to question like "how" and "why", i.e. the same type of questions as for experiments. The results from case studies are harder to interpret as the control over the environment in which the study is conducted is very small. It is thus important to plan a case study carefully to make it possible to generalize the results.

Another way to distinguish between different types of results from research is based on the type of research that has lead to them. This has been expressed for human computer interaction by Brooks [27] and adapted for software engineering by Shaw [18]. The need for this clarification is due to the tension between research results that are limited in scope but backed up by experiments showing statistical basis for conclusions and broader results that are based on observations that are more difficult to validate. 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 where this is done 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.

2.2 Case Study Methodology

The use of case studies requires careful planning and execution. This section describes a structured way of performing case studies in software engineering research. As our knowledge regarding how to conduct a proper case study has evolved throughout this research, parts of the methodological thinking described

here are missing or are less formal in the included research papers. It is thus necessary to ensure that future case studies are carried out with great care and methodology awareness.

To successfully perform a case study, several steps are needed as described in [25]. The first step is to prepare a case study design. Based on the design, the data collection is planned, the evidence collected and analyzed. Finally, the result from the case study is reported.

A case study design must include enough information to provide guidance for the execution. It can be compared to a project plan and need to include a study question, a proposition, a description of the unit of analysis, a description of how to link the collected data to the proposition and criteria for how to interpret the data. The definition of the *study question* is important as it guides and directs the research. Both the substance (what is the study about) and the form (the type of question) are important. The *proposition* describes a possible answer to the question and is sometimes formulated as a hypothesis. The *unit of analysis* describes the case, i.e. what should be studied. The possibility to select a proper unit of analysis depends highly on the selected proposition. A well formulated proposition makes it easier to focus on the right object for the study. To simplify the analysis of data collected in a case study, the process for how the data is to be *linked to the proposition* should be described when designing the study. It is closely related to the definition of *criteria for interpreting the data*. As the case study is designed it is advisable to ensure that different types of threats to the validity of the results are addressed. This is preferably done in a review of the case study design and documented in a plan.

When the case study has been designed, it is possible to plan for the data collection. The plan should contain the needed skills for the researchers involved, training and preparation for the data collection, development of a case study protocol, screening of case study candidates and testing the preparation in a case study pilot.

The data can be collected in several ways including document reviews, interviews and observations. The methods complement each other and can also be used to corroborate findings, i.e. ensure that observations are consistent and validated through the occurrence of several sources.

After the data has been collected it should be analyzed using the methods described in the case study plan. Using the criteria for data interpretation, conclusions regarding the proposition are drawn. Finally the data is reported, either for a specific case study or a collection of studies.

2.3 Selected Research Strategy

The purpose of this research is to produce *findings* responding to *why* some companies are able to perform product integration without encountering problems while others fail. The aim is also to understand *how* they achieve success or failure.

This makes it necessary to carefully select an appropriate strategy, i.e. one that can provide results from which valid generalizations can be made. Since the research is performed in an industrial setting, experiments are difficult to perform as the environment cannot be controlled. This resulted in a decision to base the research primarily on case studies.

The overall research strategy selected for this thesis consists of three steps that have been repeated in a number of iterations. First, a number of questions were formulated based on existing methods, models and theories. These questions are related to methodologies for collecting relevant information regarding processes in product development organizations and to the efficiency and effectiveness of product integration. The questions concerning information collection have been focused on how to collect reliable data from an organization. The questions regarding the product integration process are related to practices described in standards and reference models and the use of technology as means to reduce problems and increase performance.

Second, the questions were used to decide what investigations should be made on the data collection experiences available from earlier research (paper A and B) and what should be the focus in the case studies. Case studies were needed to further understand the different factors contributing to successful product integration (paper C and D). Third, the results from the investigations, case studies and additional analyses have lead to conclusions supporting or refuting existing theories or models, and helped to further elaborate the research questions.

The description of the approach is idealistic and in reality, the process was iterative. The initial questions were vague and had to be expanded as well as clarified through early findings. Also, there were ideas about the expected findings and conclusions from the start.

The case studies have been directed towards process and technology. To get coverage of the different sides of product development, also the *people* aspect needs to be investigated. The chosen approach was to analyze the effect of the application of different ethical theories on the actions performed in product development. This facilitates the understanding of one additional type of influence on the results in product development, with specific focus on product integration.

The advantages, but also the difficulties in this research lie in the fact that it is based in an industrial setting. Doing research in industry enables us to work with research questions originating in reality. The drawback is the complexity in doing generalizations and in distinguishing the influence from the structures investigated from other factors. Using reasoning, theoretical replication and rival theories [25], an attempt is made to isolate the aspects under study. The details of the methods used in each case can be found in each paper.

2.4 Research Questions

The key question for this research is how to increase the efficiency and effectiveness when putting software pieces together into something of expected quality in time. In this context efficiency means that the activities conducted with an optimal effort in the right way. Effectiveness means that efforts are concentrated on the activities that give the right output from the process. Indications from investigations performed in industry have shown that the use of component-based technologies can be assumed to help in increasing the efficiency and effectiveness when integrating software products [4]. This leads to the main question for this thesis:

Q: To what degree will the use of component-based technology assist in increasing the efficiency and effectiveness of the product integration process for software products?

To be able to respond to this question, a number of steps need to be taken and investigated. The first step is related to the collection of data. A clear understanding of how data can be collected from an industrial setting is crucial. Organizations are sometimes sensitive for investigations interfering with the development projects. To ensure that proper information about processes from product development organizations is collected, there is a need to understand what type of investigation can be made. The first question to be investigated is hence:

Q1: What are the possibilities to use non-intrusive methods for investigations of product development processes?

The proposition in response to this question is that it may be possible to perform a number of smaller investigations to achieve results that are reliable and valid enough to base improvement plans on. To investigate this further, it is necessary to understand when it is appropriate to use different types of evaluations. This leads to the next question:

Q2: What are the characteristics of an organization that can guide the selection of an appropriate evaluation method?

The proposition is that openness (i.e. the willingness to embrace external assistance in improving internal processes) and maturity are two important characteristics and through the analysis of the improvement work in five development units, it is concluded that a scheme based on these two attributes can help in selecting the right type of evaluation model.

The next question is directly derived from the main research question:

Q3: How well can the practices described in a specific standard be expected to reduce problems encountered in the integration of products?

Our investigations give at hand that the types of observed problems in product integration can be reduced through following the practices described. It was also noted that the practices described in standards and models very often are collected

from lessons learned, and not validated through research, i.e. the practices can be described as rules-of-thumb. Indications that component based technology assist in reducing the problems were also observed, and this leads to Q4.

Q4: To what degree can synergies between the use of a specific technology and the best practices in product integration be expected?

The case studies support the idea that the problems experienced in a product development organization map to the practices described in a reference model. It was also concluded that there are indications that a specific technology, component-based technology, can help an organization to follow the model and through that achieve improvements in the performance of product integration.

There are also other factors influencing the effectiveness and efficiency in the product integration process [10]. Among these are the decisions on what ethical principles should be followed in an organization or selected by the individual engineers.

Q5: How will the selection (explicit or implicit) of different ethical principles affect the efficiency and effectiveness of the product integration process?

It is proposed that the selection of ethical principles will have an effect on the development results and through a theoretical analysis the consequences for different moral frameworks are suggested.

We have concluded that through following the available good practices from standards and models, development organization can avoid problems in the product integration process. The use of component-based technology and how this can reduce the problems has been explored. In addition we propose that the moral framework selected by an individual influences the development processes. This also implies that support from technology to follow good practices would be beneficial. However, as a consequence of the increased understanding of the integration process and the influence from using component based technology we arrive with a new major research question which is an elaboration of the original question:

Q (Derived): To what degree can component based technology assist organizations and individuals in following validated good practices in product integration?

The main focus in the research presented in this thesis has been on *Question 3* and *Question 4*, with the intention to find factors that can be investigated to find improvement potential in the product integration process.

3 RESEARCH RESULTS AND CONTRIBUTIONS

"Nothing shocks me. I'm a scientist."

Indiana Jones

This chapter describes the research results, the contribution and a discussion about validity. Section 3.1 organizes the results per paper, while section 3.2 summarizes the contribution.

3.1 Summary of Included Papers

The papers include in this thesis cover three different areas; methods for collecting data from a product development organization, descriptions of different case studies regarding product integration and reasoning about the influence of different ethical directions on product development.

Paper A: Are limited Non-intrusive CMMI-based Appraisals Enough?

> **Abstract.** An integral part of the strategy for performance improvement within the product development at ABB is the use of CMMI-based appraisals. Each appraisal represents an investment by the organization to lay the best possible foundation for improvements. The challenge is to balance the investment, the intrusiveness and the benefits. Depending on different organizational characteristics, different kinds of appraisals should be used. All appraisals are driven by data collection and consequently the quality of an appraisal depends on the data collection methods used. In this paper we outline strategies used in ABB for selection of appropriate CMMI appraisals and data collection methods. Early results indicate that the use of a series of appraisals can be a way to overcome the resistance in an organization. We also claim that a discussion is needed on the reliability and validity of the appraisal methodologies and on the feasibility to base decisions regarding process improvement strategies on appraisal results.

In Proceedings of the ESEIW 2003 Workshop on Empirical Studies in Software Engineering WSESE 2003, Rome, Italy, September, 2003

Authors: **Stig Larsson**, Fredrik Ekdahl

The present author's contribution was in the description of good practices in product integration, related work, the case study, as well as parts of the analysis and conclusions.

Paper B: Selecting CMMI Appraisal Classes Based on Maturity and Openness

> **Abstract.** Over the last eight years, different approaches have been used to diagnose the performance in ABB organizations developing software. The

efforts build to a large degree on methods from the Software Engineering Institute (SEI). In this paper we examine the experiences from five organizations through a description of the pathways that we have observed in the maturity development. We also propose a way to classify organizations based on two organizational characteristics, maturity and openness. Based on this classification, a simple method for the selection of how to collect performance data from the organizations is described.

In PROFES 2004 – 5th International Conference on Product Focused Software Process Improvement, Kansai Science City, Japan, April, 2004

Authors: **Stig Larsson,** Fredrik Ekdahl

The present author's contribution was in the description of good practices in product integration as well as parts of the case study, analysis, and conclusions.

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

Abstract. The expectations for a well working integration process are described in the Capability Maturity Model Integration (CMMI). Often during the integration process, weaknesses of the entire development process become visible. This is usually too late and too costly. Particular development processes and use of particular technologies may help to improve the performance of the integration process by providing proper input to it. For example, by the use of a component-based approach, the development process changes. Some of these changes may help in performing according to the process expectations. In this paper, examples of problems that have been observed in the integration process are described. Through a case study we describe a number of practical problems in current development projects. Based on this case study, we analyze how a component-based approach could help and lead to a more effective integration process.

In Euromicro Conference, Rennes, France, August, 2004

Authors: **Stig Larsson,** Ivica Crnkovic, Fredrik Ekdahl

The present author's contribution was in the description of good practices in product integration, methodology, the case study, as well as parts of the analysis and conclusions.

Paper D: Case Study: Software Product Integration Practices

Abstract. Organizations often encounter problems in the Product Integration process. The difficulties include finding errors at integration related to mismatch between the different components and problems in other parts of the system than the one that was changed. The question is if

these problems can be decreased if the awareness of the integration process is increased in other activities. To get better understanding of this problem we have analyzed the integration process in two product development organizations. One of the organizations has two different groups with slightly different integration routines while the other is basing the development on well defined components. The obstacles found in product integration are highlighted and related to best practices as described in the interim standard EIA-713.1. Our conclusion from this study is that the current descriptions for best practices in product integration are available in standards and models, but are insufficiently used and can be supported by technology to be accepted and utilized by the product developers.

Accepted to PROFES 2005, Oulu, Finland, June, 2005

Authors: **Stig Larsson,** Ivica Crnkovic

The present author's contribution was in the description of good practices in product integration, methodology, the case study, as well as parts of the analysis and conclusions.

Paper E: Expected Influence of Ethics on Product Development Processes

Abstract. Product development efficiency and effectiveness is depending on a process being well executed. The actions of individuals included in the processes are influenced by the ethical and moral orientations that have been selected by each individual, whether this selection is conscious or not. This paper describes different ethical choices and the expected effect they may have on the development process exemplified by the product integration process for software products. The different frameworks analyzed are utilitarianism, rights ethics, duty ethics, virtue ethics and ethical egoism. Our conclusion is that the adherence to specific moral frameworks simplifies the alignment of actions to the practices described in product development models and standards and supports through this a more successful execution of product development projects. This conclusion is also confirmed through a comparison between the different directions and several codes of ethics for engineers issued by organizations such as IEEE as these combine features from several of the ethical directions.

Technical report, a shorter version has been accepted to ECAP-2005, Västerås, Sweden, June, 2005.

Author: **Stig Larsson**

3.2 Contribution

The main contributions in the thesis are the following:

- *Investigation and analysis of the possibilities to use non-intrusive evaluation methods to determine the need for process improvement.*

Paper A describes different effects and consequences of the use of different types of evaluation methods, concentrating on investigation techniques that are non-intrusive. This relates to *Q1*, and proposes on how companies can set up a strategy using lightweight evaluation methods as a basis for identifying improvement areas.

- *Analysis of the effects of maturity and openness in process improvement diagnostics and a proposed model for selection of appraisal class.*

The proposed model for selecting investigation method described in paper B is based on observations of how improvement work has been performed in several different development organizations over a number of years. The proposed model is a response to Q2, and also includes a proposed scale for classifying the openness of an organization.

- *Assessment of practices described in standards and models and the practices used in industry resulting in a description of what specific practices help in reducing problems in product integration.*

An analysis of the three development groups from two different organizations described in paper D provides evidence supporting that the adherence to the best practices as described in a standard can help in reducing problems in product integration. This in combination with the similar results from paper C validates in part the statement that the basic practices described in the investigated models and standards are increasing the effectiveness and efficiency of product integration. Also, in both papers we document the indications that the use of a specific technology, component based software, help the organizations to follow the described practices is described. These contributions connect to Q3 and Q4.

- *Further understanding of the process of product integration and the possibilities to use component-based technologies to assist in improving the integration processes.*

Additional understanding of the product integration process has been documented in papers C, D and E. In these we provide evidence that support that the efficiency and effectiveness of the product integration process depend on several factors such as maturity, technology used and morale in the organization. From the case and literature studies it is clear that the three factors we have chosen, process, technology and people, all influence the performance of product integration in the development organization. Also the interaction between these three aspects can be observed in the investigations. This relates to Q3, Q4 and Q5.

To conclude: Our main research questions are to what degree the use of component-based technology can contribute to increased efficiency and effectiveness in product integration for software products, and how this also can help in following best practices. Our research demonstrates that following the best practices described in standards and models increases the possibility of successful product integration, and that the use of component-based technology helps in achieving this. However, additional research is needed to be able to generalize these findings.

3.3 Validity discussion

There is in all research a need to understand if adequate validity is achieved. The validity must be related to the extent of the generalization that is the target of the research. The scope of our research covers the development of industrial systems with substantial part software. The teams are fairly small in all organizations, between 5 and 50 development engineers working in each project. In addition, the investigations cover only small geographical area, Sweden. All the organizations are part of large global companies which leads to a target of the generalization to be small and medium-sized projects located in Sweden developing industrial software in global companies. Different measures have been taken to reduce the threats to validity, both in the individual case studies and in the overall research design. This section summarizes the actions, and details can be found in each of the papers.

Four types of validity threats have been considered in this research [25]. Construct validity relates to the data collected and how this data represent the investigated phenomenon. Internal validity concerns the connection between the observed behavior and the proposed explanation for this behavior. The possibilities to generalize the results from a study are dealt with through looking at the external validity. Finally, the reliability covers the possibilities to reach the same conclusions if the study was repeated by another researcher.

The **construct validity** is dealt with through the investigations regarding data collection methods and through multiple sources for the data in the case studies. The investigations of data collection methods were made to ensure that also non-intrusive assessment methods give reliable results, The sources for data in the case studies where interviews of people having different roles such as developer, project manager, integration responsible, and line manager as well as project documentation and quality system documentation. In addition, the researchers experience in software product development provided a basis for relevant focus of the investigations and interviews.

The **internal validity** has been handled through explanation building and observations over several years for data collection methods. The connection between the observations of what practices are performed and the effect of the performance in the case studies has been made through matching collected data with theoretically predicted events. The collected data includes empirically observed events

documented in interviews and document reviews. The theoretically predicted events in the case studies are for example problems that are expected to appear in the projects if practices are not followed.

The **external** validity is dealt with through the use and description of several case studies covering different development organizations performing different practices. The cases have been selected from the same application area; software for industrial use, and from the same type of organizations, small and medium-sized projects in global companies operating in Sweden. This leads to the possibility to observe an analytical replication [25]. One organization performing specific tasks in a process has no problems in that process while another organization, not performing the tasks, has problems. Further studies with an expanded set of development organizations from other application areas and from other geographical areas would increase the external validity as this would eliminate additional factors as possible causes for the findings. The external validity can never be proven, but its accuracy can be increased by observing the same patterns in these additional cases, or by reaching the same conclusions using different research methods.

The **reliability** of the study has been handled through description of data collection methods and the creation of a research data base including background material, case study preparation material and data collected in the case studies. The description of data collection methods and case study preparation material will enable similar investigations to be made to increase the possibilities to make a wider generalization. The collected data includes notes from interviews and from document reviews as well as material collected from quality system and project presentations. This will enable other researchers to investigate the material to ensure that proper analysis has been made and that valid conclusions have been drawn.

One additional aspect related to validity is to what extent the conclusions are useful and complete. Our main conclusion that best practices help in reducing the problems in product integration is an answer to our main research question. However, this answer has limitations. One example is the question of emerging system properties. The best practices in standards and models only cover this area implicitly, and the relationship with the explicit descriptions of what to do for successful product integration is complex. The requirements on system properties need to be transferred to component requirements, but are difficult or impossible to verify on the component level. These types of questions are not sufficiently addressed by best practices although they are important for the final results. Consequently, our conclusion is limited as it only states that it is useful, neither that it is sufficient nor even necessary to follow best practices to be successful in product integration. Still the results have a research value; they belong to classes of improvements by building experiences of best practices, a known method important when other methods are not feasible.

4 PRACTICES FOR SOFTWARE PRODUCT INTEGRATION

"If you can't describe what you are doing as a process, you don't know what you're doing."

W. Edwards Deming

This chapter includes a short description of a selection of standards and models that are used in industry today. It also includes a comparison between the practices and the observations from the case studies in paper C and D.

4.1 Practices in Standards and Models

Both standards and models, two different types of reference material, have been considered in this section. The included standards and models are typically used by product development organizations to obtain a common language, to ensure that the development performed covers necessary activities, to guide improvement activities and to show compliance. The selection of standards and models in this section is based on available information from standardization organizations such as ISO [28], and IEEE [29] and references from organizations such as SEI [30] and SCCI [31]. Two additional selection criteria have been used in the choice of standards and models. The first was that the standard or model should be relevant to product development of products that include software. The second criterion was that the standard or model should include requirements on product integration, implicitly or explicitly. The descriptions include the purpose and intention of the standards and models and details regarding the product and software integration processes included. It should be noted that efforts are made to harmonize several of these standards and models such as IEEE Std 1220-1998 [32], EIA-632 [13] and ISO/IEC 15288 [15]. It remains to be seen what format the description of product integration will have in the harmonized material.

For each standard and model, a table summarizes the described product integration practices and the adherence to the tasks as observed in the cases described in paper C and D. The actions and tasks are summarized as practices even if the specific standard or model uses a different terminology. Note that these summaries are for information purposes only, and that the original text in the standards and models should be used for any implementation. The four columns describing the results from the case studies include: an indication for each case if the practice has been observed (+), not observed (-), not investigated (?) or too generic to be determined (G), and if there are indications of problems connected to the practice (*).

Two issues limit the value of this analysis. The first is that all the indications have been based on the material available from the case studies. As no explicit coverage of all the practices described in each standard and model was made in the cases, practices may be performed even if no evidence can be found in the material. The second issue is that problems may exist in the organization without indications in the

table, again based on the fact that not all practices where explicitly covered in the case studies.

Based on an analysis of the case study data and the practices, two observations have been made. The first is that problems encountered in the case studies can in all cases except one be related to practices described in standards and models, but not performed by the organization or project. The second observation is that not all standards and models have practices to which the problems can be related. This can be seen as an indication that there is further development needed before an established body-of-knowledge is available for the product integration area.

4.1.1 ISO/IEC 12207, Information technology – Software life cycle process

The purpose of ISO/IEC 12207 is to provide the software industry with a well-defined terminology for software life cycle processes [12]. It contains the different processes, activities and tasks that make up a software life cycle, and applies to the development, operation and maintenance of software products as well as to acquisition and supply of software products, systems and services.

ISO/IEC 12207 includes two parts related to product integration. The first is covering the integration of software units or components into software items that can be integrated into a system. The tasks described are: to develop and document an integration plan for each software item that has been identified in the system architectural design, to integrate and test the aggregates as described in the plan, to update the user documentation and to develop and document a set of tests for each requirement of the software items. The standard also lists a number of criteria that should be used for evaluation of each work product developed in the software integration process as well as a requirement to conduct joint reviews.

The second part describes the system integration tasks. These are: to integrate the software into the system and to test the requirement of the system. There is also a list of criteria for evaluation of the integrated system.

The practice to which we can relate some of the problems found in the case studies is to ensure that the integrated software is ready for verification. This standard has no requirements on the handling of interfaces, which represents the cause of many of the problems found in the case studies. Our conclusion is that the standard does not fully cover the needs for effective product integration. The requirement that the user documentation should be updated has not been investigated in the case studies.

Table 1. ISO/IEC 12207:1995 compared to cases

Adapted description of practice in ISO/IEC 12207:1995	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Develop and document an integration plan for each software item	+	-	+	-
Integrate and test the aggregates as described in the plan	+	-	+	-
Update the user documentation	?	?	?	?
Develop and document a set of tests for each requirement of the software items	+	+	+	+
Ensure that the integrated software item is ready for verification	- *	- *	- *	+
Integrate the software item into the system	+	+	+	+
Test the requirement of the system	+	+	+	+

4.1.2 IEEE Std 1220-1998, Application and Management of the Systems Engineering Process

IEEE Std 1220-1998 [32] provides guidelines for product development organizations to ensure that the products resulting from development activities are properly designed to be affordable to produce, own, operate, maintain and dispose of with appropriate consideration for the health and environmental risks.

Product integration is described in section 5.4.1, "System integration and test". Assembling and integration of subcomponents are to be done progressively into complete components, components into assemblies, assemblies into subsystems and subsystems into products. Also the combination of products, processes and services into a system is described. At each level of assembly and integration, it is expected that sufficient testing is performed to ensure operational effectiveness, usability, trainability, interface conformance, conformance with specified requirements, producibility, and supportability.

In our analysis, we have found the descriptions in IEEE 1220-1998 to be of insufficient detail to allow verification that each step in the process is performed. The steps and actions described in other standards and models are not visible in this standard, except in an implicit way.

Table 2. IEEE 1220-1998 compared to cases

Adapted description of practice in IEEE Std 1220-1998	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Ensure that combining lower-level elements results in a functioning and unified higher-level element	G	G	G	G
Satisfy logical and design interfaces	G	G	G	G

4.1.3 EIA-632

The purpose of the EIA-632 standard [13] is to provide developers with fundamental processes that assist in engineering a system. In this context, a developer can be an enterprise or an organization. The use of the standard should help developers to develop requirements that enable delivery of system solutions in a cost-effective way, delivering within cost, schedule and risk constraints and to provide a system that satisfies the different stakeholders over the life-cycle of the products that make up the system.

The integration of parts into products is included in the requirement for implementation. The implementation practices include expectations, that the developers should plan for and execute tasks such as validating the subsystems received for assembling and assembling validated subsystem products into the test items or end products to be verified.

The requirements in EIA-632 are concrete, but do not include requirements in all the areas where we have found problems in the case studies. Specifically, the handling of interfaces is not explicitly mentioned.

Table 3. EIA-632 compared to cases

Adapted description of practices in EIA-632	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Plan for validation of subsystems and assembling of subsystems	+	-	+	-
Validate subsystems to be assembled	- *	- *	- *	+
Assemble the validated subsystems	+	+	+	+

4.1.4 Capability Maturity Model Integration (CMMI), Version 1.1

The Capability Maturity Model Integration, CMMI, from the Software Engineering Institute describes what is considered as best practices for product and systems engineering [9]. The model includes process areas covering the full product life cycle for the development and maintenance of products and services. The purpose of the model is to provide a basis for process improvement, and includes guidelines for how to select improvement areas.

For each of the process areas described in CMMI, a purpose is described. For Product Integration it is "to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product". It is detailed in three goals which are supported by a total of nine practices that are specific for product integration. The goals are: Prepare for product integration, Ensure interface compatibility and Assemble product components and deliver the product.

All problems encountered in the case studies regarding product integration can be related to practices that are described in the CMMI. However, there are also a few practices that have not been performed by the units without causing problems. This leads to the conclusion that there needs to be further investigations and validations to ensure that requirements in the model are not only sufficient, but also necessary. One example is that in two cases presented in paper D, no integration sequence determined, but no problem could be related to this.

Table 4. CMMI compared to cases

Adapted description of practice in CMMI	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Determine integration sequence	+	-	+	-
Establish the product integration environment	+	+	+	+
Establish product integration procedures and criteria	- *	- *	+	-
Review interface descriptions for completeness	- *	-	-	-
Manage interfaces	- *	-	-	-
Confirm readiness of product components for integration	- *	- *	- *	+
Assemble product components	+	+	+	+
Evaluate assembled product components	+	+	+	+
Package and deliver the product or product component	+	+	+	+

4.1.5 EAI-731.1

The purpose of the interim standard EIA-731.1 is to support the development and improvement of systems engineering capability [14]. The standard is structured to support different activities performed to improve the performance in a development organization such as appraisals, process improvement, and process design.

Product integration is described in the section Integrate System which describes practices connected to product integration strategy, interface coordination, integration preparation and system element integration.

As with CMMI, all problems found in the case studies can be related to practices in EIA-731.1. There are also practices in the standards that are not performed, but we have not been able to identify any relations between these and the observed problems.

Table 5. EIA-731.1 compared to cases

Adapted description of practice in EIA-731.1	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Develop an integration strategy	?	+ *	+	+
Document the strategy as a part of an integration plan, and develop an integration plan early in the program	+	-	+	-
Coordinate interface definition, design and changes between affected groups	- *	- *	-	+
Review interface data and ensure complete coverage	- *	-	-	-
Verify receipt of components in accordance with architecture	- *	- *	- *	+
Verify that the interfaces comply with the interface documentation prior to assembly	- *	- *	+	+
Coordinate the receipt of system elements for system integration according to the planned integration strategy	-	-	+	-
Assemble aggregates of system elements in accordance with the integration plan	+	+	+	+
Checkout assembled aggregates of system elements	+	+	+	+

4.1.6 ISO/IEC 15288, Systems engineering – system life cycle processes

ISO/IEC 15288:2002 is intended to describe the life cycle of systems [15]. The standard is to be applied to the full life cycle of systems from inception, development, production, utilization, and support to retirement of the system. It is noted in the standard that the implementation typically involves a selection of a set of processes applicable for the project or organization.

Product integration is described in the section Integration Process. The purpose with this process is to assemble a system that is consistent with the architectural design. System elements should be combined to form partial or complete products. The activities includes definition of a strategy for integration, identification of design constraints based on the strategy, preparation of facilities that enable the integration, reception of validated system elements in accordance with a schedule and the actual integration. In addition, there is a requirement to store information about the integration into an appropriate database.

ISO/IEC 15288:2002 introduces a requirement that the constraints from the integration strategy on design should be identified. This requirement is not represented in any of the other standards, and is not investigated in the case studies. However, we believe this is an important area that needs to be further investigated as it is closely related to the requirements on how interfaces are handled.

The standard covers most of the problems found in the case studies, but are only implicitly covering parts of the interface handling. All problems found in the case studies can therefore not be related to a specific practice in this standard.

Table 6. ISO/IEC 15288:2002 compared to cases

Adapted description of practice in ISO/IEC 15288:2002	Paper 3, Case 1	Paper 4, Case 1	Paper 4, Case 2	Paper 4, Case 3
Define an assembly sequence and strategy that minimizes the system integration risks	+	+	+	+
Identify the constraints on the design arising from the integration strategy	?	?	?	?
Obtain integration enabling systems and specified materials according to the defined integration process	+	+	+	+
Obtain system elements in accordance with agreed schedules	- *	?	?	?
Assure that the system elements have been verified against acceptance criteria specified in an agreement	- *	- *	+	+
Integrate system elements in accordance with applicable interface control descriptions and defined assembly procedures, using the specified integration facilities	+	+	+	+
Record the integration information in an appropriate database	-	-	-	-

4.1.7 ISO 9000, Quality management systems

The ISO 9000 family of international quality management standards and guidelines is often used as a basis for establishing quality management systems [33]. It builds on eight principles; customer focus, leadership, involvement of people, process approach, system approach to management, continual improvement, factual approach to decision making, and mutually beneficial supplier relationships. All these principles are important for product integration.

The requirements on a quality management system are specified in the ISO 9001:2000 standard. Section 7.3 in 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 limited and not mentioned explicitly, this standard has not been further analyzed.

4.2 Comparison between Practices in Different Standards and Models

Table 7 summarizes the product integration process as described in different standards and models and provides a basis for comparison. The descriptions of practices have been made generic and form a combination of the practices described in each standard and model and can be used as a guideline for the definition of a product integration process. However, if the purpose is to implement a standard or a model, the original texts should be used. The three different types of indications in the table are if the practice is explicitly described in the standard (E), implicitly described (I) or not described (-). The implicit description may be through a generic statement that the type of activity should be performed, or that it can be interpreted as being included in another requirement.

This comparison shows that the content in the standards and models is expanding; additional practices are added and already existing practices are made more precise. The expectations on the preparation for integration and the handling of interfaces have been made more explicit over time. The conclusion is that there is an on-going development of the area and an increased agreement on what can be considered to be best practices. Additional investigations and comparisons are in progress to understand how the area evolves, what factors are determining what is added to the standards and models and if there are specific considerations that should be made for different types of products and systems. There is also a need to validate the changes that are made through case studies in different types of product development organizations.

Table 7. Product integration process in selected standards and models

	ISO/IEC 12207	IEEE Std 1220-1998	EIA-632	CMMI	EIA-731.1	ISO/IEC 15288
Publication date	Aug 1995	Dec 1998	Jan 1999	Mar 2002	Aug 2002	Nov 2002
Generic description of activity						
Define an integration strategy	-	I	I	I	E	E
Develop an integration plan based on the strategy	E	I	E	E	E	E
Define and establish an environment for integration	-	I	I	E	-	E
Define criteria for delivery of components	I	I	I	E	I	E
Define interfaces	-	I	I	E	E	I
Review interface descriptions for completeness	-	I	I	E	E	E
Ensure coordination of interface changes	-	I	I	E	E	I
Review adherence to defined interfaces	-	I	I	E	E	E
Develop and document a set of tests for each requirement of the assembled components	E	I	E	I	-	I
Verify completeness of components obtained for integration through checking criteria for delivery	E	I	E	E	E	E
Deliver/obtain components as agreed in the schedule	E	I	I	I	E	E
Integrate/assemble components as planned	E	I	E	E	E	E
Evaluate/test the assembled components	E	E	E	E	E	E
Record the integration information in an appropriate repository	-	-	-	-	-	E
Package and deliver the product or product component	I	-	I	E	-	-
Update the user documentation	E	-		-	-	-

5 CONCLUSION

"'It seems very pretty,' she said when she had finished it, 'but it's rather hard to understand!'"

Lewis Carroll

Investigations in industry indicate that the integration of products with significant part software needs to be improved. The research presented in this thesis supports these indications for products developed for industrial applications. The research demonstrates also that the existing descriptions of practices for product integration in standards and models help in achieving successful integration if applied, and that the use of component-based software can simplify this. We have also seen that there is a continued development of the description of best practices in the product integration process area. The results from this research can be generalized to small and medium projects in global organization developing software in Sweden for industrial applications. A broader generalization requires additional case studies in other environments and a deeper comparison of different case studies and complementary approaches such as surveys and theoretical model building. The challenge for the future is to continue the development towards an agreed body-of-knowledge for the product integration area. There is a need to further investigate the reasons for the lack of use of proven good practices, and to understand why the implementation of product integration practices sometimes fails.

Several different additional directions for future research have been identified in this thesis. We have seen indications that the selection of a specific technology increases the ability to do efficient and effective product integration. Additional organizations using different technologies should be investigated and compared to clarify the dependencies. A related direction is to look at the influence architectural decisions have on product integration.

Methods for how to determine the best improvement proposals for product integration for different types of organizations should be investigated, enhanced and possibly developed. This probably requires an agreed body-of-knowledge for product integration that supports different types of organizations, and the use of different development models. The standards and models investigated in this thesis do not prescribe specific development models, but the selection is likely to influence the ability to follow the practices and to be successful in the product integration.

The research described in this thesis has helped the author to an increased understanding of how case studies need to be executed to give reliable results. The need for careful planning, concise research questions, well formulated propositions; and an understanding of how to increase the validity through alert selection of investigation methods and appropriate case will augment future research accomplishments.

REFERENCES

[1] IEEE, Guide to the Software Engineering Body of Knowledge, 2001,http://www.swebok.org, (link valid April 2005).

[2] IEEE, Guide to the Software Engineering Body of Knowledge, 2004,http://www.swebok.org, (link valid April 2005).

[3] Zeidler, C., Componentware Glory and Crux for early industrial adopters, Object Oriented Programming conference OOP 2000, Munich, Germany, 2000.

[4] Winter, M., C. Zeidler and C. Stich, "The PECOS Software Process", Workshop on Components-based Software Development Processes, ICSR 7, Austin, TX USA, 2002.

[5] Stallinger, F., B. Henderson-Sellers and J. Torgensson, "The OOSPICE Assessment Component: Customizing Process Assessment to CBD", in Business Component-Based Software Engineering, edited by F. Barbier, Kluwer Academic Publishers, Boston, USA, 2002.

[6] Myers, G.J., *The Art of Software Testing,* John Wiley, New York, 1979.

[7] "The Economic Impacts of Inadequate Infrastructure for Software Testing", RTI, National Institute of Standards and Technology, Gaithersburg, MD, USA, May 2002

[8] Campanella, J., editor, *Principles of Quality Costs: Principles, Implementation, and Use*, 3rd edition, ASQ Press, ISBN 0-87389-443-X, Milwaukee, WN, USA, 1999.

[9] Chrissis, M.B., M. Konrad, S. Shrum, *CMMI*, Addison-Wesley, Boston, MA, 2003.

[10] Kellner, M.I., J.W. Over, "A Software Quality Improvement Framework," Proceedings from Software Engineering Forum, Olivetti Information Services, Milan, Italy, 1992.

[11] Land, R., I. Crnkovic, "Existing Approaches to Software Integration - and a Challenge for the Future", Fourth Conference on Software Engineering Research and Practice in Sweden, Linköping, Sweden, October, 2004

[12] ISO/IEC 12207:1995, "Information technology – Software life cycle processes", ISO/IEC 1995.

[13] ANSI/EIA-632-1999, "Processes for Engineering a System", Government Electronic and Information Technology Association, Electronic Industries Alliance, 1999.

[14] EIA-731.1, "Systems Engineering Capability Model", Electronic Industries Alliance, 2002.

[15] ISO/IEC 15288:2002, International Standard, "Systems engineering – Systems life cycle processes", ISO/IEC 2002.

[16] http://evidence.cs.keele.ac.uk/rebse.html, Link to information about Workshop on Realising Evidence-Based Software Engineering (REBSE), affiliated with the 27th Int'l Conf on Software Engineering, 2005. (Link valid April 2005.)

[17] Zelkowitz M. V., D. Wallace, "Experimental validation in software engineering", Information and Software Technology, 39, Elsevier, 1997.

[18] Shaw, M., "What makes good research in software engineering?," International Journal of Software Tools for Technology Transfer", vol. l4., no.1, 2002.

[19] Oivo, M., P. Kuvaja, P. Pulli, J. Similä,"Software Engineering Research Strategy: Combining Experimental and Explorative Research", Proceedings of the 5th International Conference, PROFES 2004, Kansai Science City, Japan, LNCS 3009, Springer 2004.

[20] Shaw, M., "The coming-of-age of software architecture research," Proceedings of the 23rd International Conference on Software Engineering, IEEE Computer Society, 2001.
[21] Basili, V. R., R. W. Selby, D. H. Hutchens, "Experimentation in Software Engineering", IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, 1986
[22] Kitchenham B. ; Pickard L. ; Pfleeger S.L, "Case studies for method and tool evaluation", IEEE Software, Vol: 12, Issue: 4, 1995
[23] Zelkowitz, M. V., D. Wallace, "Validating the Benefit of New Software Technology", Software Quality Practitioner 1.1, 1998
[24] Tichy, W. F., P. Lukowicz, L. Prechelt, E. A. Heinz, "Experimental evaluation in computer science: A quantitative study", The Journal of Systems and Software, 1995.
[25] Yin R. K., *Case Study Research: Design and Methods* (3rd edition), ISBN 0-7619-2553-8, Sage Publications, 2003
[26] Wohlin, C., P- Runesson, M. Höst, M. C. Ohlsson, B- Regnell, A. Wesslén, *Experimentation in Software Engineering, An Introduction,* ISBN 0-7923-8682-5, Kluwer Academic Publishers, 2000
[27] Brooks, F. P., "Grasping Reality Through Illusion – Interactive Graphics Serving Science", Proceedings of the SIGCHI conference on Human factors in computing systems, Washington D.C. United States, 1988
[28] ISO, International Standardization Organization, http://www.iso.org/. (Link valid April 2005.)
[29] IEEE, The Institute of Electrical and Electronics Engineers, http://www.ieee.org/. (Link valid April 2005.)
[30] SEI, Software Engineering Institute, http://www.sei.cmu.edu/. (Link valid April 2005.)
[31] SCCI, Systems and Software Consortium, http://www.software.org/ssci/. (Link valid April 2005.)
[32] IEEE Std 1220-1998, "IEEE Standard for Application and Management of the Systems Engineering Process", IEEE 1998.
[33] ISO 9001:2000, International Standard, "Quality management system – Requirements", ISO 2002.

PAPER A:

ARE LIMITED, NON-INTRUSIVE CMMI-BASED APPRAISALS ENOUGH?

Stig Larsson and Fredrik Ekdahl
In Proceedings of the ESEIW 2003 Workshop on Empirical Studies in Software Engineering WSESE 2003, Rome, Italy, September 2003

Abstract

An integral part of the strategy for performance improvement within the product development at ABB is the use of CMMI-based appraisals. Each appraisal represents an investment by the organization to lay the best possible foundation for improvements. The challenge is to balance the investment, the intrusiveness and the benefits. Depending on different organizational characteristics, different kinds of appraisals should be used. All appraisals are driven by data collection and consequently the quality of an appraisal depends on the data collection methods used. In this paper we outline strategies used in ABB for selection of appropriate CMMI appraisals and data collection methods. Early results indicate that the use of a series of appraisals can be a way to overcome the resistance in an organization. We also claim that a discussion is needed on the reliability and validity of the appraisal methodologies and on the feasibility to base decisions regarding process improvement strategies on appraisal results.

1 Introduction

ABB, a global operator in power and automation technologies, has been developing industrial software products for more than 30 years. Today, steps are taken to transform ABB into an organization recognized for its software product development excellence. Key to this transformation is the use of the CMMI (Capability Maturity Model Integration) [1][2] and its companion IDEAL^{SM} model [3] for organizational improvement, both developed by the Software Engineering Institute (SEI).

In this paper, the challenges facing an organization using the CMMI to diagnose performance will be highlighted. The aim is to fuel the discussion on the reliability and validity of CMMI appraisals and how they can be improved.

In the second section of this paper, the relationships between maturity, capability and performance are discussed and the connection to CMMI established. Section three and four describe the models and data collection methods used within ABB to manage performance improvement in product development organizations. In section

five, we discuss how to choose the right appraisal class. Section six describes the results so far from ABB. Finally, section seven describes the topics for discussion.

2 Maturity, Capability and Performance

Maturity represents an organization's ability to consistently follow and improve its processes. An increase in maturity is driven by improved process capability. In turn, process capability can be best described as the variability of the expected results from a process. Improved process capability gives greater predictability and increased performance of the process. Finally, performance represents the result that is actually achieved by the process.

The CMMI was developed to answer the need for structured improvement of software product development organizations, and the model itself is derived from extensive empirical data. The CMMI is based on five maturity levels, each representing an evolutionary stage that organizations pass through as they increase in maturity. Each maturity level consists of a set of carefully selected Process Areas of relevance to the specific evolutionary stage. In this way, the levels provide an implicit prioritization of which processes to address during each evolutionary stage.

Figure 1 illustrates a few of the Process Areas in the CMMI and their interconnections. Each Process Area consists of a set of Goals and a set of corresponding Practices. The Technical Solution Process Area develops the product components and the necessary product component data that is later used by the Product Integration Process Area to integrate the final product that is delivered to the customer. Both the Technical Solution and the Product Integration Process Areas rely on the Verification and the Validation Process Areas to continuously ensure that product components meet specified requirements and fulfill the expectations of the customer.

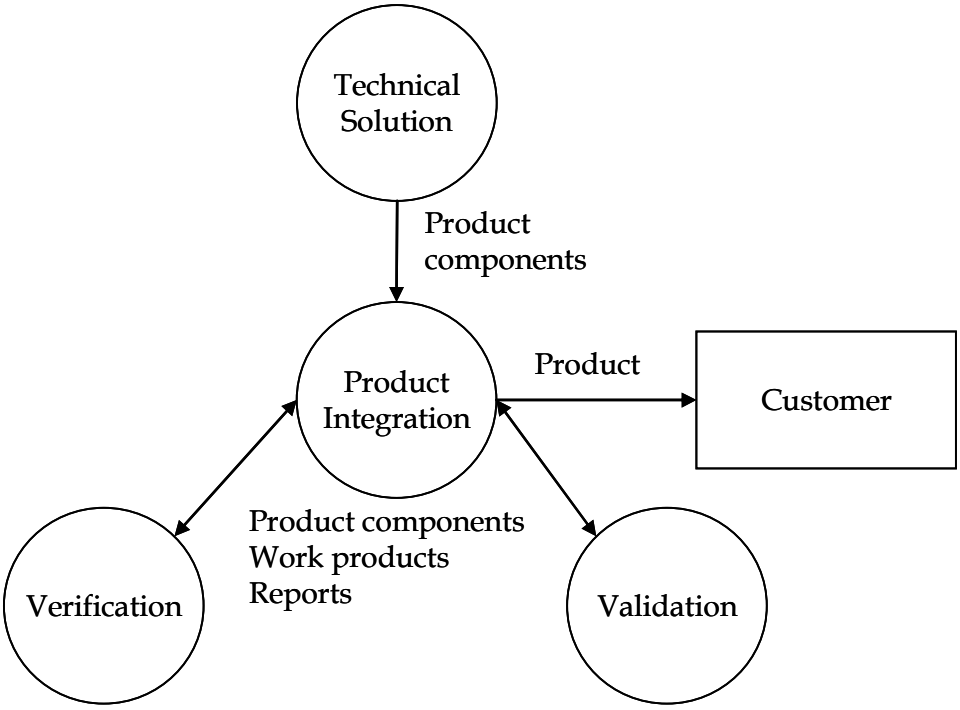

Fig. 1. Sample Illustration of Process Areas from the CMMI

3 The ABB IDEAL Model

The ABB IDEAL Model has been developed to serve as the recommended work model for initiating, planning, executing, reviewing and evaluating performance improvement activities in product development. It is reminiscent of the IDEAL Model developed by the SEI [3], which in turn is based on the Plan-, Do-, Check-, and Act Cycle [4][5].

The ABB IDEAL, shown in Figure 2, consists of five phases; Initiate, Diagnose, Establish, Act and Leverage. Each of the phases serves a specific purpose in an improvement effort, but here attention will only be given to the Diagnose phase.

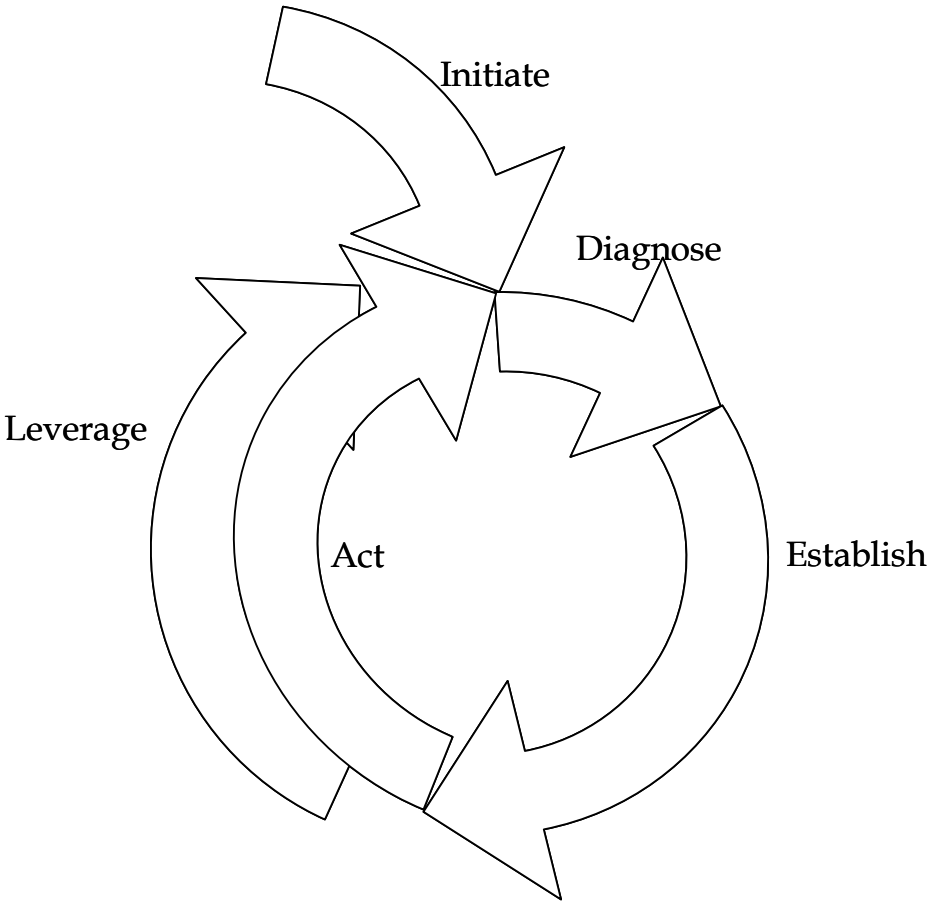

Fig. 2. Phases of the ABB IDEAL Model

The purpose of the Diagnose Phase is to baseline current level of performance against a selected reference model, such as the CMMI, and to identify the most important areas for improvement. This includes planning, execution and follow-up of appropriate appraisal activities. Findings from the appraisal activities are used as a basis for identifying appropriate improvement actions that are then the focus of the Establish, Act and Leverage phases.

The Appraisal Requirements for CMMI (ARC 1.1) [6] defines three classes of appraisals (Class A, B and C). All three classes are used in ABB and they all display different strengths and weaknesses [7]. The three classes of appraisals can roughly be categorized according to cost, i.e. the investment made by the organization to conduct the appraisal and intrusiveness, i.e. how great an interference with ordinary operations the appraisal represents.

As appraisal findings are fundamental to the subsequent activities it is of utmost importance that they show high reliability and validity. Consequently reliability and validity represents and additional way to categorize appraisals. Here, reliability represents the ability to produce findings that are relevant irrespective of variations

in sources of data and validity represents the ability to pinpoint the most relevant findings.

Class A appraisals are the most comprehensive, but require substantial resources and may be considered very intrusive by the organization being appraised. For example, a Class A requires an authorized lead assessor and at least three different data collection methods, including onsite interviews.

Class B appraisals are less comprehensive and consequently less intrusive, but still require considerable resources. A Class B does not require an authorized lead assessor and only requires two different data collection methods. However, onsite interviews are still required.

Finally, Class C appraisals are the least comprehensive, but again require fewer resources and are less intrusive. A Class C can be done remotely, as onsite interviews are not required. Also, only one data collection method is required.

The comprehensiveness of the appraisals of course influences the reliability and validity of the appraisal results. Table 1 summarizes the characteristics of the different classes of appraisals.

Table 1. Characteristics of different appraisal classes

Class of appraisal	**A**	**B**	**C**
Cost	High	Medium	Low
Intrusiveness	High	Medium	Low
Validity	High	High	Low
Reliability	High	Medium	Low

4 Data Collection Methodology

One of the primary driving forces of reliability and validity of an appraisal is the data collection methodology used. In ABB, four data collection methods are used in the appraisals; process mapping, questionnaires, document reviews and interviews. The choice of data collection methods depends on the appraisal class chosen.

To illustrate the four different methods for data collection and the kind of data that is obtained, we will use the Product Integration process area as an example. As shown in Figure 1, Product Integration relates to several other process areas and requires communication and documentation in the project to be of high quality to ensure effective execution. (Refer to [1] for a detailed description of the Product Integration process area.)

The purpose of process mapping is to graphically capture the current state of the product development process in the appraised organization. It covers all process areas involved from requirement capturing to delivery to customer. The mapping is done in cooperation between the appraisal team and a representative of the organization, and typically happens during the planning phase of the appraisal, as the map will greatly facilitate any additional data collection. Process mapping gives a good overview and allows identification of weak or missing practices as well as unnecessary complex process flows. However, as the number of individuals involved is small, the result may be biased. For the Product Integration process area, the mapping reveals what interfaces exist towards activities in other process areas and the absence of activities expected in the process.

The second method for data collection is questionnaires. The questionnaires consist of a set of standard questions for each of the process areas in scope. Questionnaires allow large organizational coverage, i.e. a large number of respondents, which provides for a good estimate of the level of understanding of the process in the organization. However, even if the possibility to add comments is a part of the questionnaire, the questions are seldom open ended and there is really no possibility for follow-up questions, which could give additional data.

The third method for data collection is document reviews during which documents and other work products resulting from process execution are analyzed. Document reviews allow the appraisal team to review evidence of the claims, about the existence, content and quality of documents, made in questionnaires and in interviews.

Document review for the Product Integration process area would normally include integration plans that typically show the integration steps and strategies for integration testing, and also the requirements on other parts of the project through expected delivery dates and functionality. Also, interface lists showing how well the project has defined connections between different parts of the product, integration acceptance records, product integration reports and delivery documents displaying the state of the handover between different parts of the project and the organization are likely to be reviewed. According to the CMMI, all these documents, or the fact that they do not exist, indicate how well the Product Integration process is performed.

The fourth and final data collection method is interviews. Interviews are conducted with members of the organization selected based on their anticipated knowledge of how the process is actually performed. Although the interviews are highly structured, i.e. follow a predetermined format, it is important that they are perceived as quite informal by the interviewees. This requires careful planning and well-trained interviewers. Much as the questionnaires, the interviews are based on a standard set of questions, but now there is more room to explore follow-up questions and give the interviewees greater possibility to describe the process in their own words. Of

course, interview sessions can only reveal the individual perception of the interviewees on process execution and organizational adherence.

For the Product Integration process area, highest priority should be given to individuals responsible for the execution of the product integration. This group can provide information on how the process is performed and on how different development groups deliver components for integration. They are also likely to provide proposals for how the process can be improved. Also, individual developers in the component development groups, technical writers etc., and the receivers of the result (product verification, validation, production groups) are high priority interviewees to get a complete view of the process. These groups can describe what requirements come from the product integration function. Additional candidates for interviews include project and line managers, configuration managers and requirement managers.

5 Choosing an Appraisal Class

The selection of a specific appraisal class largely determines what type of data collection methods can be used. In a Class C appraisal, it is not required to use more than one data collection method. This means that the data might be incomplete. For example, when using only questionnaires to investigate the Product Integration process area, there might be activities, such as checking for interface compatibility with interface specifications, which are performed but not documented as expected in the way the questions are written. This could result in the erroneous observation.

Class B appraisals always include interviews and at least one more data collection method. Through the interviews, findings in the other data collection method can be confirmed and additional information be found that better reflects the activities performed. In the Product Integration example, the interface compatibility check would be found in the discussions with the product integration responsible and corroborated through interviews with the component development groups.

To secure reliability and validity of the results from the appraisal, a Class A must be used. In a Class A, at least three data collection methods are required. This will enable crosschecking between different groups, but also to verify the findings through other sources such as protocols from interface compatibility reviews.

There are two main reasons for not always selecting Class A appraisals; the openness and maturity of the organization. Openness represents the willingness of an organization to accept the costs of an appraisal, to accept the inconvenience of external examination and to make a genuine effort to improve. The maturity of an organization influences the possibilities to conduct more thorough appraisals. That is, mature organizations are more likely to appreciate and benefit from the results from an appraisal.

The openness and maturity of an organization can thus guide the organization to the appropriate class of appraisal.

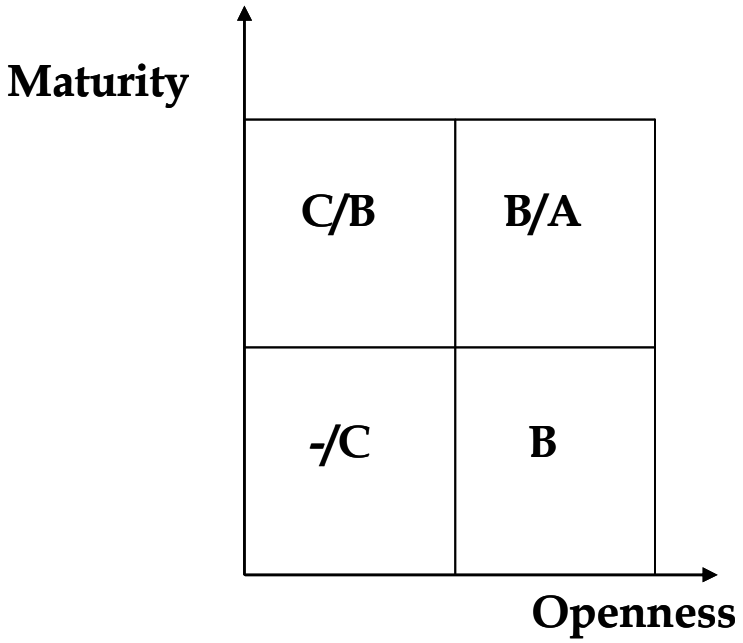

Fig. 3. Selection of appraisal class based on organization characteristics

Figure 3 illustrates what class of appraisal should, in our experience, be selected for an organization with specific characteristics. Organizations with low maturity need experience in process improvement before exhaustive appraisals give the expected benefits. Organizations that are sensitive to intrusive appraisal methods need confirmation that external assistance in finding improvement opportunities is useful.

For organizations with low maturity that are sensitive to external appraisals, very basic processes may have to be put in place before any appraisal is helpful. As the organization develops process knowledge and the first appraisal is conducted, the unit will get an understanding of what the result of an appraisal can be and start to appreciate the external view. For these organizations, class C appraisals are appropriate.

If an immature organization is open to external assistance in finding improvement opportunities the appraisal can be extended to cover a larger part of the organization. By increasing scope and through using additional data collection methods, more reliable results will be available. A class B appraisal supports this.

Also mature organizations may need results to accept comprehensive appraisals. Reviews and audits made by external organizations are very often considered as an inspection or a test that needs to be passed by the unit that is examined. Frequent reviews can have the side effect that the organizations become sensitive to external interference in the process improvement work. This means that for less open

organizations, less intrusive appraisal methods should be selected to build confidence. It may be necessary to start with Class C appraisals or to use a class B appraisal on a limited set of process areas in a limited part of the organization.

Also for the open and mature organization, the use of Class A appraisals can be enhanced with more frequent class B appraisals to verify the direction of the performance improvements in the organization.

6 Early Results and conclusion

The approach used by ABB is to classify the organization with respect to maturity and openness and select an appropriate roadmap. A common solution for the initial work is to perform several Class C appraisals to capture urgent issues followed by a Class B appraisal to secure the quality of improvement progress. Class C appraisals cover a limited number of process areas in each appraisal. The appraisals are scoped so that a broad coverage of process areas is obtained over time. Class B appraisals, on the other hand, are scoped to achieve complete coverage of a set of process areas.

The concept of combining Class B and C appraisals is attractive also to less mature organizations, as the intrusiveness is relatively small, but still allows continuous observation. The results from the combined series of appraisals provide a longitudinal perspective enabling continuous control of the process improvement activities. The use of a series of appraisals can be a way to overcome the resistance in an organization. Although initial results from using this approach are promising, further study is needed to validate its feasibility for broad application.

7 Topics for Discussion

ABB has chosen to use CMMI appraisals as the tool to study the progress of process improvement activities. Consequently it is vital for achieving our targets that this tool meets our purpose to produce reliable and valid findings in a cost effective and less intrusive way. Therefore, a discussion is needed on the reliability and validity of appraisal methodologies and on how they can be further improved.

In addition, a discussion is desirable on the feasibility of our approach as a basis for decision-making regarding process improvement roadmaps. Are limited, non-intrusive CMMI appraisals really enough?

References

[1] CMMI® Product Development Team, "CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1), Staged Representation", Technical Report CMU/SEI-2002-TR-012, Pittsburgh, PA (2002)

[2] CMMI® Product Development Team, "CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1), Continuous Representation", Technical Report CMU/SEI-2002-TR-011, Pittsburgh, PA (2002)

[3] McFeeley, R., "IDEAL℠. A User's Guide for Software Process Improvement", Handbook, CMU/SEI-96-HB-001, Pittsburgh, PA (1996)

[4] Deming, W. E., *Out of the Crisis*, Cambridge University Press, Cambridge, MA (1986)

[5] Deming, W. E., *The New Economics - For Industry, Government, Education*, Massachusetts Institute of Technology Cambridge, MA (1993)

[6] CMMI® Product Development Team, "ARC, V1.1; Appraisal Requirements for CMMI, Version 1.1", Technical Report CMU/SEI-2001-TR-034, Pittsburgh, PA (2001)

[7] Minnich, I., "CMMI Appraisal Methodologies: Choosing What Is Right for You", Crosstalk, Feb 2002, http://www.stsc.hill.af.mil/crosstalk/2002/02/minnich.html (link valid April 2005).

PAPER B:

SELECTING CMMI APPRAISAL CLASSES BASED ON MATURITY AND OPENNESS

Stig Larsson, Fredrik Ekdahl
In PROFES 2004 Conference, Kansai Science City, Japan, April 2004

Abstract.

Over the last eight years, different approaches have been used to diagnose the performance in ABB organizations developing software. The efforts build to a large degree on methods from the Software Engineering Institute (SEI). In this paper we examine the experiences from five organizations through a description of the pathways that we have observed in the maturity development. We also propose a way to classify organizations based on two organizational characteristics, maturity and openness. Based on this classification, a simple method for the selection of how to collect performance data from the organizations is described.

1 Introduction

Considerable effort has been put into transforming ABB into an organization recognized for its software development excellence. This is in line with a strategic redirection of operations towards primarily the software intensive process automation market. Since the mid-nineties, several performance improvement initiatives have been run on a national level. Tangible results are evident in many of the participating organizations. Today, there is a global program for performance improvement in product development that coordinates improvement activities throughout ABB.

Using structured process improvement methods is a well-documented path towards increased maturity in product development organizations. In this paper, we adhere to the Software Engineering Institute's [1] definition of maturity as an organization's ability to consistently follow and improve its processes. In ABB, we have over the last eight years tried a number of approaches in different parts of the organization. Our focus has been on software development units, primarily developing software intensive products. Due to organizational dynamics and management short-term focus, some of the initiatives have been disrupted or slowed down. Also, the expected development towards higher maturity and accompanying results in quality and development speed has not been received. As organizations probably will continue to be dynamic and the focus on short-term results occasionally will re-appear, we need to better understand how performance improvement can be achieved in spite of these and similar circumstances.

Based on our experiences in ABB we introduce the concept of evolutionary paths in maturity and use five short case studies from organizations within ABB to illustrate how organizations develop over time in terms of maturity. As a structure for the illustration of how these organizations have evolved, we use a simple model that allows a longitudinal perspective. The model defines four different types of organizations, each exhibiting a few unique characteristics. Based on our experiences within ABB, we also propose strategies for approaching each of the four types of organizations. Future work will include more comprehensive evaluations of different approaches for the different organizational types. This will eventually lead to the possibility to provide strong guidance for all types of organizations relative to diagnostic strategies and to improve the maturity in software product development.

In ABB, the Capability Maturity Model Integration (CMMI, [1][2]) is used as the preferred process reference model and that is reflected in this paper. However, we would like to point out that the proposed classification and the corresponding strategies are valid independently of the reference model used.

The CMMI (and its predecessor the Software Capability Maturity Model, SW-CMM) was developed to answer the need for more structured and long lasting improvement of software product development organizations. Both the SW-CMM and the CMMI are derived from extensive industry experience.

The CMMI consists of five maturity levels, each representing an evolutionary stage that organizations pass through as they increase in maturity. Each maturity level consists of a set of carefully selected Process Areas of relevance to the specific evolutionary stage. This way, the levels provide an implicit prioritization of which processes to address during each evolutionary stage. Each Process Area consists of a set of Goals and a set of corresponding Practices.

The CMMI is a process reference model and it does not contain any explicit support for how to actually achieve improvement. Therefore the SEI developed the IDEAL model [3], which in detail describes how to use the CMMI (or in fact the SW-CMM) to professionally improve the maturity of an organization.

The IDEAL Model consists of five phases; Initiate, Diagnose, Establish, Act and Leverage, each serving a specific purpose in an ongoing improvement effort.

The purpose of the Diagnose phase is to baseline current level of maturity against a selected reference model, such as the CMMI. This includes planning, execution and follow-up of appropriate appraisal activities. For other reference models the terms audits or assessments are often used, which corresponds to the term appraisal used here. Findings from the appraisal activities are used as a basis for identifying appropriate improvement actions that are then the focus of the Establish, Act and Leverage phases.

The remainder of this paper is organized as follows. Section two describes how improvements in product development maturity can be based on appraisals. We also

define the characteristics selected for classification of organizations. Section three contains the five case studies that illustrate the evolutionary pathways that we then use as a basis for the different diagnostic strategies detailed in section four. The paper ends with some conclusions and a brief look into possibilities for future work.

2 Appraisal-Driven Improvement

There is strong support in the literature that conducting diagnostics activities, i.e. systematically identifying strengths and weaknesses in an organization, contributes to the advancement of organizational excellence [5]. Diagnostic activities come in different forms, including for example appraisals, audits, assessments and reviews. Common to all are that the results can be used as the foundation for future improvement activities.

In ABB, the preferred diagnostics methodology is CMMI appraisals. An appraisal is an examination of one or more processes that an organization does to and for itself for the purposes of process improvement. It is conducted by a trained team of professionals using an appraisal reference model as the basis for determining strengths and weaknesses [1]. The Appraisal Requirements for CMMI (ARC 1.1) [4] defines three classes of appraisals (Class A, B and C). All three classes are used in ABB and they all display different strengths and weaknesses [6]. Class A appraisals are the most comprehensive, but require substantial resources and may be considered very intrusive by the organization being appraised. Class B appraisals are less comprehensive and consequently less intrusive, but still require considerable resources. Finally, Class C appraisals are the least comprehensive, but again require fewer resources and are less intrusive. The comprehensiveness of the appraisals of course influences the reliability and validity of the appraisal results. Table 1 summarizes the characteristics of the different classes of appraisals.

Table 1. Appraisal class characteristics

Class of appraisal	**A**	**B**	**C**
Size of appraisal team	8-10	3-4	1-2
Appraisal time	10 days	3-4 days	1-2 days
Minimum # of data collection methods	3	2	1
On-site interview required	Yes	Yes	No
Cost	High	Medium	Low
Intrusiveness	High	Medium	Low
Validity	High	High	Low
Reliability	High	Medium	Low

As appraisal findings are fundamental to the subsequent activities it is of utmost importance that they show high reliability and validity. In this context, reliability represents the ability to produce findings that are relevant irrespective of variations in sources of data and validity represents the ability to pinpoint the most relevant findings.

There are several organizational characteristics that improve the chances of successful improvement activities. [7] provides an overview of characteristics of particular importance for software process improvement based on case studies and experience reports from 56 different organizations.

Similarly, there are several organizational characteristics that contribute to truly effective appraisal activities. [8] contains a good overview of what is needed from an organization to fully benefit from an appraisal. Among the more important can be mentioned:

Strong management commitment: Improvement does not happen overnight. It is important the senior management is willing and patient enough to visibly stand behind a genuine improvement effort based on the appraisal results. This also includes allowing and withstanding the scrutiny that appraisal activities entail.

Resources for Performance Improvement: Financial as well as human resources are necessary to cover the appraisal costs and of course also to fund subsequent improvement activities.

Improvement Infrastructure: In order to effectively make use of appraisal results, a certain degree of organizational structure must be available. It is for example common to establish an engineering process group that is responsible for coordinating the improvement activities. Also a process champion that reports directly to the management team on improvement progress is very valuable.

Organizational characteristics such as these are fairly easily identified when observing an organization from the outside. Consequently, they can be used as indicators of the readiness of an organization to work effectively with appraisal-driven and professional improvement.

In our contacts with development units within ABB we have identified two characteristics that distinguish the organizations from each other relative to the effectiveness of different classes of appraisals. The first characteristic is Maturity, i.e. the ability of an organization to follow and improve its processes. We have found that the maturity of an organization influences the possibilities to conduct appraisals effectively, as well as the ability to appreciate and benefit from the results from an appraisal. The concept of maturity according to the SEI's definition can be considered fairly objective as there are established techniques and considerable experience in the field of maturity evaluation.

The second characteristic is Openness, which basically captures the inclination of an organization to embrace external help. In this paper, openness is an aggregate that represents the willingness of an organization to accept the costs of an appraisal, to accept the inconvenience of external examination and to make a genuine effort to improve. This definition of openness is influenced by the list of organizational characteristics identified in [8]. A proposed scale, ranging from 1 to 5, for the openness of an organization:

5. Management and organization are openly requesting assistance in performance improvement, including appraisals, improvement planning and subject matter expertise.
4. Management and organization accept assistance in performance improvement, including appraisals and improvement planning.
3. Management accepts assistance in conducting appraisals.
2. Management accepts discussing performance improvement.
1. No access to management or organization.

We acknowledge the dangers of using a characteristic with such obvious subjectivity. However, we still argue its relevance for guiding appraisal activities, as will be illustrated in the case studies and in our recommendations.

There are of course many organizational characteristics beside maturity and openness that also influence the potential benefit and success of appraisal activities. For example, the size or complexity of he organization, the current lifecycle phase of key projects, or current financial status. We do not claim to be in any way exhaustive in our search for suitable organizational characteristics. Instead, we have settled with a set of characteristics that appears good enough for our purposes.

3 Case Studies

The case studies include five different ABB organizations. The size of the organizations, in number of software developers, range from 60 to 120 and all five organizations develop software that is used in industrial environments. We have refrained from revealing actual maturity levels in the case studies, primarily because they add no value to the discussion or the conclusions of this paper, but also as they are considered confidential and proprietary to the individual organizations. Note that the overall goal of the case organizations is not to achieve a certain maturity level, but to improve performance.

3.1 Research Method

Each organization has been examined over a period of several years with different data collection methods. Appraisal and assessments have been made using the SW-CMM or the CMMI as a reference model. Typically, class B appraisals or equivalent assessments have been made. Support work ranges from leading workshops and providing training to assisting the organizations in developing and institutionalizing

new processes. Through moderating peer reviews where software development managers meet to review each other's improvement activities, additional information has been collected. Finally, networks of software development managers have been organized. Both authors of this paper have been directly involved in all of the above activities. This allows us to estimate maturity and openness. Maturity estimates have been based on data available from appraisals, assessments and audits. Openness has been estimated based on the scale presented above.

The long-term observations have given us the possibility to describe a maturity development path for each organization. The extended time for the observations as well as the use of the diversity of the data collection methods should increases the reliability of the collected data.

Through the methods used in this study, we conclude that this is a hermeneutic research endeavor. Thus, we need to acknowledge the influence of our research on the units and the results. We claim that although we influence the different pathways, this influence does not change the validity of the observations or the conclusions regarding the classification of the different organizations.

We acknowledge the relatively unstructured data collection method used when determining the openness. This is also true for some of the maturity observations. The data collection is thus partly subjective. As a result, the possibility to replicate this particular study is small. However, we consider our observations as a good starting point for conducting a more stringent investigation.

In addition, through the long-term observations and extensive experience from the organizations, we still claim that the observations are reliable enough to qualitatively determine the openness and maturity for the organizations. In addition to this, the use of external observers doing peer review of our research adds validity.

3.2 Organization A

Organization A develops real time control systems and tools for adaptation of the system for different applications. The unit has developed software for almost 30 years, but has been exposed to repeated changes in the organizational structure over the last 15 years.

Figure 1 shows the development of the maturity and openness over the last eight years.

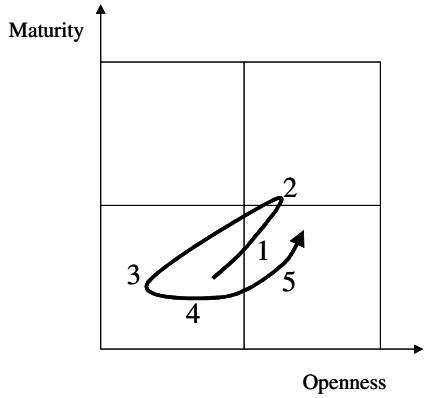

Fig. 1. Organization A maturity pathway

The following development has been observed:

1. *Use of CMM to improve performance*. The decision to base the improvement efforts on SW-CMM was firmly committed with the management. During this period, the performance improved as well as the maturity. Also, the need to involve external experts for specific areas was acknowledged.
2. *Major change in commitment*. Due to acquisitions and new development management, the commitment to increase maturity as a means to further increase performance was lost. Consequently, the openness diminished.
3. *Organizational changes*. As a result of the insufficient results, clarification of global responsibilities where made, again giving the responsibility to the local management to drive improvement.
4. *High pressure to deliver*. Improvement efforts were in this period not prioritized. However, the openness increased as a result of local management commitment to improvement efforts.
5. *Initial results from process improvement activities*. Through the commitment and openness, we can now observe initial results in maturity.

3.3 Organization B

With the experience of software development spanning over more than 20 years, this organization has established the basic routines for product development. The real time part of the system is primarily operating as an independent product, but is more and more connected to a larger system. The unit is also expanding the software development to PC based products. This expansion has partly been made through acquisitions.

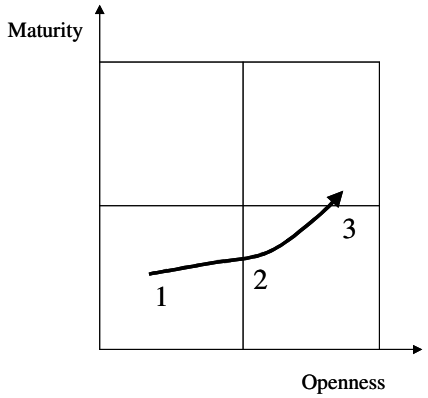

Fig. 2. Organization B maturity pathway

The development for the last five years for organization B is shown in figure 2:

1. *Yearly improvement plans introduced*. The efforts to improve were based on targets for the organizations. However, no diagnostic activity was performed.
2. *Increased pressure to improve timeliness and quality*. The pressure lead to increased acceptance to involve external resources in finding improvement areas.
3. *Introduction of CMMI*. Recently the organization has decided to use CMMI as a tool for identifying weakness and initiating improvements in projects.

3.4 Organization C

Through several reorganizations and mergers, this organization has managed to maintain focus on the products that are entirely software based. The product development is primarily directed towards the evolution of the product platform. The organization is also cooperating with similar units around the world. However, there is currently no requirement that the products from different units should be integrated.

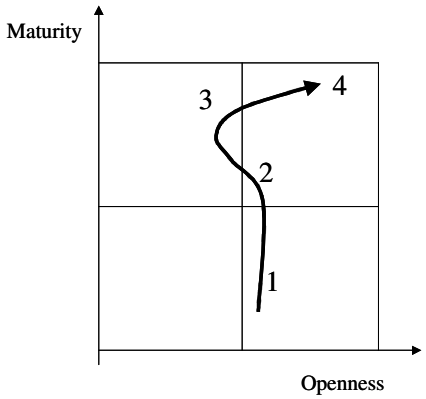

Fig. 3. Organization C maturity pathway

The pathway shown in figure 3 describes the development for the last three years:

1. *Strategic decision to use SW-CMM*. The decision was a response to market requirements and included the intention to use SW-CMM as a diagnostic tool as well as a roadmap for improvements. External expertise was requested from start.
2. *Initial results achieved*. As results where observed, the organization started to work more in isolation.
3. *Acceptance that external assistance is beneficial*. As the pace of the improvements slowed down, a more open attitude could be observed.
4. *Preparation for class A appraisal in progress*. The current status is that a class A appraisal is planned.

3.5 Organization D

The organization develops products with a tight integration of hardware and software. The real time requirements on the system are tough and also one discriminating factor in the marketplace. The development of software has gradually grown over the last 15 years.

As shown in figure 4, the organization has developed in the last five years as follows:

1. *SW-CMM used for improvement*. Through a Class A type of assessment, the organization started an improvement effort.
2. *Internal improvements*. As the organization was maturing, the strategy was to decrease external involvement in the efforts to diagnose and improve performance.
3. *Organizational changes*. The organization was transferred to belong to a different part of ABB. This resulted in a change in senior management with less interest for improvement efforts.
4. *Change agent changes*. In addition to organizational changes, the unit has had several changes in the staff responsible for processes and improvement efforts.

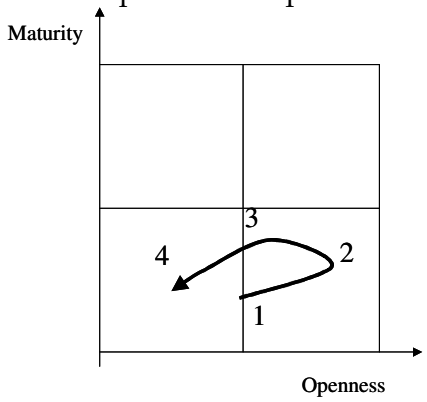

Fig. 4. Organization D maturity pathway

3.6 Organization E

The organization develops real time products with a tight integration of hardware and software. The reliability requirements on the products are high. The dependence on the software part of the product has steadily grown over the last 25 years.

Figure 5 shows how the organization has developed over the last eight years:

1. *Increased demands on performance*. As the products started to include more and more software, the structure and complexity grew. This led to the needs to improve. The organization started to gradually improve through internal projects.
2. *External support*. As the organization matured, an appreciation of external assistance started to grow.

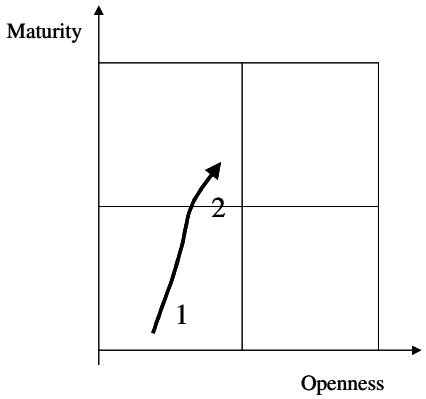

Fig. 5. Organization E maturity pathway

3.7 Longitudinal Perspective

The development of maturity over time may also add to the understanding of how to choose a diagnostic strategy. Figure 6 shows this development for the development organizations in the case studies.

We draw three additional conclusions from our experiences; to build up maturity takes considerable time, to build up the confidence for external support also takes time and finally, both maturity and confidence can easily and quickly be lost.

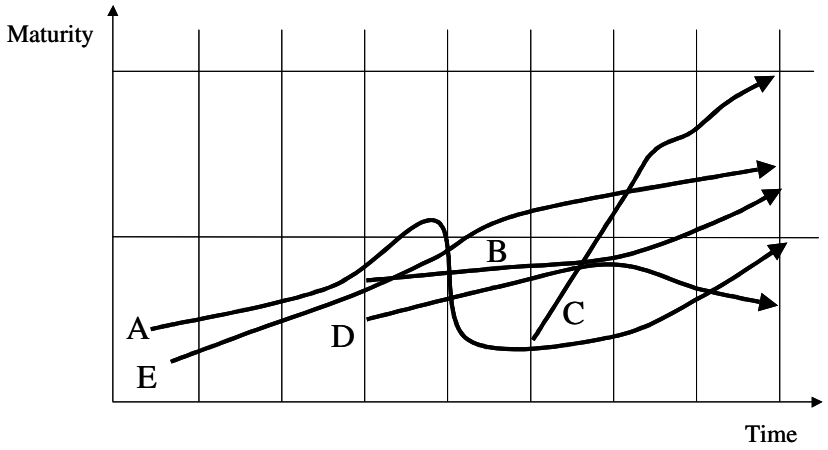

Fig. 6. Longitudinal maturity development for organizations A through E

4 Diagnostic Strategies

Diagnostic activities are key to the overall success of an improvement activity. Consequently, choosing an appropriate diagnostic strategy largely determines how effective an improvement effort will be. In an attempt to capture the experiences we have made in ABB and to allow some amount of generalization we propose classifying organizations based on their openness and maturity. In addition we propose a primary diagnostics strategy for each type of organizations.

4.1 Organizational Classification

From the observed case studies, we have identified four types of organizations as shown in Figure 7. Type 1 organizations are immature organizations that are unwilling to let external experts help in improvements. Typically these organizations think they are more mature than is the case.

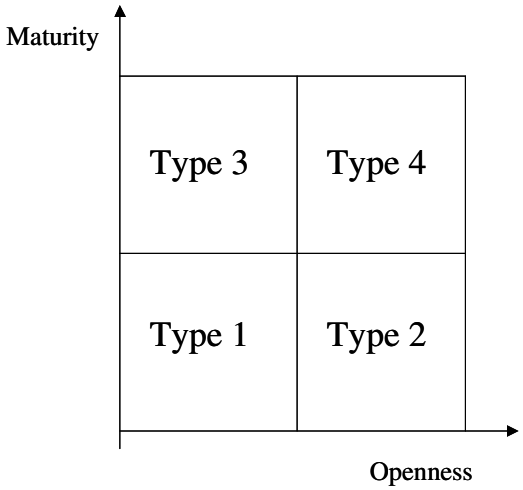

Fig. 7. Organization classification based on openness and maturity

In Type 2 organizations, where openness is found but the maturity is low, an acceptance that the organization is immature is often found. There is often awareness in this type of organization that external assistance is needed.

Type 3 organizations very often have a tradition of internal process improvement that has led to a mature status. The lack of openness in these organizations can have several different reasons, but as it may slow down or even stop the improvement activities, efforts should be made to overcome it.

Finally, Type 4 organizations take full advantage of the external expertise and use that to maintain their maturity.

4.2 Recommended Diagnostic Activities

Organizational openness and maturity are both relatively easy to observe. They are also highly indicative of the kind of external support and organization is ready for. Consequently, the openness and maturity of an organization can be used to guide decision making in the development of an improvement strategy. This is especially true when choosing appropriate diagnostic activities.

Figure 8 illustrates the recommended diagnostic methods, expressed in different classes of CMMI appraisals, for each of the four types of organizations. Our recommendation is based on a combination of the observations made in the case studies and other organizations within ABB and externally.

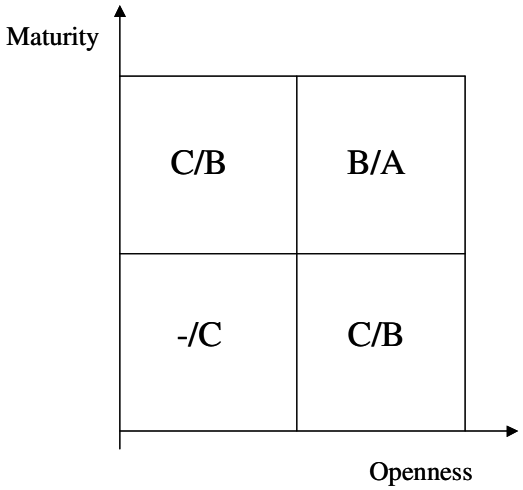

Fig. 8. Selection of CMMI Appraisal Class (A through C) based on organization characteristics

The basic observation is that organizations with low maturity need experience in process improvement before exhaustive appraisals give the expected benefits. In addition, organizations that are sensitive to intrusive appraisal methods need confirmation that external assistance in finding improvement opportunities is useful.

For organizations with low maturity that are sensitive to external appraisals (type 1 organizations), very basic processes may have to be put in place before any appraisal is helpful. As the organization is immature, the area to improve should be easy to find through initial discussions with the organization. It is very important that the efforts give quick payback, as this will encourage the organization to continue the efforts. As the organization gains experience and the first appraisal is conducted, the unit will get an understanding of what the result of process improvement activities can be. The organization will also start to appreciate the view from external sources. This type of organizations typically selects Class C appraisals. Since a class C appraisal only covers a part of the organization, the selection of projects is important. Our experience is that central projects should be selected. Based on the status of these projects, the process areas to be examined are decided.

If an immature organization is open to external assistance in finding improvement opportunities (type 2 organizations) the appraisal can be extended to cover a larger part of the organization. By increasing scope and through using additional data collection methods, more reliable results will be available. A Class B appraisal supports this.

Also mature organizations may need results to accept comprehensive appraisals (type 3 organizations). Reviews and audits made by external organizations are very often considered as an inspection or a test that needs to be passed. Frequent reviews can have the side effect that the organizations become sensitive to external interference in the process improvement work. This means that for less open

organizations, less intrusive appraisal methods should be selected to build confidence. It may be necessary to start with Class C appraisals or to use a Class B appraisal on a limited set of process areas in a limited part of the organization.

When an organization is both open and mature (type 4 organizations), the use of Class A appraisals can be enhanced with frequent Class B appraisals to verify the direction of the performance improvements in the organization.

As the organizations move along a pathway, the strategy for the performance improvement needs to be adapted. We propose that the method described increases the probability for long term sustainable improvements, and can assist management in determining the appropriate level of activity in each moment.

5 SUMMARY AND CONCLUSIONS

Through the use of different diagnostic methods, we have captured and described the evolutionary paths of four organizations. To describe the development we use a mapping based on two organizational characteristics, maturity and openness. These have been selected as our experience indicates that both affect the possibilities to effectively use the results from appraisals. We acknowledge that there are several other organizational characteristics that are of equal importance for successful performance improvement. Among these are for example, organizational size, project pressure, management commitment and economical success. However, the experiences from the cases show that for the selection of the most appropriate diagnostic strategy the maturity and openness characteristics are outstanding. Based on the defined characteristics, we have identified four types of organizations, and this classification enables us to propose what class of diagnostic method should be used for a specific organization. The principle is that lower maturity organizations benefit from less intrusive appraisals, as the benefits of improving the processes need to be shown to create acceptance for external involvement in diagnostics and improvement activities. Also, if an organization is not open, the need to confirm that external assistance is beneficial.

6 FUTURE WORK

Looking into the future there are several ways to further develop the claims and recommendations made in this paper.

We would want to continue monitoring the evolutionary paths of the organizations in the case studies to get a better understanding of how organizations evolve over an extended period of time. This would in the long run enable us to identify patterns of behavior, and consequently allow prediction in some sense of how an organization is going to evolve. We would also want to extend the study to include additional organizations, as this would increase the reliability and validity of our claims.

It would also be interesting to make a comparison between the observed patterns of behavior and what can be called an "ideal" path. Often, reference models, such as the CMMI, expect and require organizations to evolve along extreme paths that are not achievable in reality. Whether or not this is detrimental for the way improvement efforts are planned an executed remains to be investigated. It is reasonable to claim that having a better understanding for more realistic evolutionary paths will help improve the way improvement efforts are set up.

It would also be interesting to further investigate the openness characteristic of organizations. In this paper, we have let openness represents an aggregate of a set of organizational traits. However, more work is needed to better understand the organizational aspects that influence the ability to improve professionally. Future study would allow development and verification of a more complete set of organizational characteristics. These characteristics could then be used as readiness indicators, much as openness is used in this paper, when developing an improvement strategy and before undertaking an improvement effort.

In turn, a more complete set of organizational characteristics, would allow identification of additional types of organizations, which would add more perspective to the analysis.

Finally, more work is needed to verify the recommendations made in this paper as to the choice of diagnostic method for different types of organizations. In the long run, it will also be possible to extend the recommendations to include not only diagnostic activities, but also the remaining phases of the IDEAL model. This would entail developing comprehensive strategies for planning and executing professional performance activities in organizations based on their unique characteristics.

REFERENCES

[1] CMMI® Product Development Team, "CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1), Staged Representation", Technical Report CMU/SEI-2002-TR-012, Pittsburgh, PA (2002)

[2] CMMI® Product Development Team, "CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1), Continuous Representation", Technical Report CMU/SEI-2002-TR-011, Pittsburgh, PA (2002)

[3] McFeeley, R., "IDEAL℠. A User's Guide for Software Process Improvement", Handbook, CMU/SEI-96-HB-001, Pittsburgh, PA (1996)

[4] CMMI® Product Development Team, "ARC, V1.1; Appraisal Requirements for CMMI, Version 1.1", Technical Report CMU/SEI-2001-TR-034, Pittsburgh, PA (2001)

[5] Kitson, D. H., and Humphrey, W. S., The Role of Assessment in Software Process Improvement, Software Engineering Institute, CMU/SEI-89-TR-3, Pittsburgh, PA (1989).

[6] Minnich, I., "CMMI Appraisal Methodologies: Choosing What Is Right for You", Crosstalk, Feb 2002, http://www.stsc.hill.af.mil/crosstalk/2002/02/minnich.html (link valid April 2005).

[7] Steltzer, D., and Mellis, W., "Success Factors of Organization Change in Software Process Improvement", Software Process Improvement and Practice, 4, (1998)

[8] Kasse, T., *Action Focused Assessments for Software Process Improvement,* Artech House Inc., Norwood, MA (2002)

PAPER C:

ON THE EXPECTED SYNERGIES BETWEEN COMPONENT-BASED SOFTWARE ENGINEERING AND BEST PRACTICES IN PRODUCT INTEGRATION

Stig Larsson, Ivica Crnkovic, Fredrik Ekdahl
In Euromicro Conference, Rennes France August 2004

Abstract

The expectations for a well working integration process are described in the Capability Maturity Model Integration (CMMI). Often during the integration process, weaknesses of the entire development process become visible. This is usually too late and too costly. Particular development processes and use of particular technologies may help to improve the performance of the integration process by providing proper input to it. For example, by the use of a component-based approach, the development process changes. Some of these changes may help in performing according to the process expectations. In this paper, examples of problems that have been observed in the integration process are described. Through a case study we describe a number of practical problems in current development projects. Based on this case study, we analyze how a component-based approach could help and lead to a more effective integration process.

1. Introduction

Product integration is a specific activity in the software development process. Very often this is also the activity where most of problems become visible and when it is either too late or at least very expensive to solve the problems. This is especially true for large and complex software products and systems which parts are developed and tested separately and when different mismatches are invisible until the products are integrated. The problems of integration usually have roots in previous phases, and most often in the lack of coordination between these phases. There are several reasons for this. First, it can be a communication problem and differences in goals between engineers conducting requirements analysis and specification, development, integration, testing and delivery of the products. Further there can be differences in the project goals (personified by project managers) and long-term goals (personified by system architects and domain experts). Second, a source of the problem is inadequate preparation of parts for the final integration. While being tested and verified on a part level, the product parts do not fit together. The reason for this problem can be inadequate test environments that are sufficient for testing particular

functions of each part in isolation, but which do not reflect the impact of a particular part on the entire product. A third source of problems is inadequate information provided from parts. Very often there are many unwritten rules and "default" assumptions known on the part level that are invalid for the whole product. A fourth type of problems is features added into particular parts that are unknown to other parts and the entire product. By adding new features (such as improvement of particular functions or protocols) the architecture of the entire system can degrade or even break down.

Many of these problems originate from the ambiguity of separations of activities in the development process. While a separation of the different parts of the development processes exists in practice, this separation is often not well defined and formalized.

In component-based software engineering (CBSE), a separation of the development of components from the product integration is one of the main characteristics [1]. This raises several questions as described in [2]: What is a component, what is included into a component specification, what are the possibilities of predicting the product properties from component properties, how does a component interact with other components and its environment and similar.

So far the research focus for component-based engineering has primarily been on technical issues, and considerably less on process issues. It is however very important to know if the development process and CBSE are synergistic; will it be more efficient and effective or will it meet new challenges and maybe unsolved problems?

In this paper our aim is to investigate what the opportunities for improvement of the integration process and the development process in general by introducing a component-based development. Can the problems described be (at least partially) solved?

To investigate this possibility our research approach is the following. From a case study of a development process that has many similarities to a component-based approach, but still is not explicitly designed so, we highlight to the main challenges and problems that become visible in the integration phase. Further we analyze these challenges and discuss the possible changes and improvements in the process by introduction of a component-based development process.

The definition of a software component used in a product follows in this paper is broad, and the term is used to describe a part of a software system. However, in the discussions regarding CBSE, the notion of a component follows to a large extent [1], i.e. software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system. We also use the definition of a product as an application that can be sold and distributed independently, and has a clear customer value on its own.

The remainder of the paper is organized as follows. Section two describes the main characteristics of the integration phase of a development process, the main characteristics of a component-based development process, the changes in the integration process implied by component-based software engineering and related work. In section three, a case study is presented to show examples of how the integration process is performed today. Section four analyzes how the use of component-based software engineering would resolve today's challenges. Finally section five contains the conclusion and proposed future work.

2. Product Integration in relation to CBSE

The product integration process for software products addresses the assembly of software components. The target is to integrate components into a product and to ensure that the product works appropriately so that it can be delivered to customers. An integration process that is working well is expected to increase the probability that a development project delivers quality products in a timely manner. Component-based software engineering is targeting similar goals; to improve the productivity through use of high-quality components with predictable behavior. This section describes these two independent methods for improving the performance in development projects, and lists possible synergies.

2.1 Product Integration Best Practices

The Capability Maturity Model Integration, CMMI, [3] defines three goals for the product integration process. These are that (i) the product integration should be prepared, (ii) interface compatibility should be ensured and that (iii) the product components should be assembled and delivered.

The preparation for product integration typically includes preparation of an integration sequence. Different integration sequences should be examined and also include test components and equipment. The established sequence should be periodically reviewed to accommodate changes in the development project. The preparation also includes the establishment of the environment needed for product integration. One important decision in the preparation of the integration environment is if it should be developed in-house or bought from outside. In practice, the system will include both components that are bought and that are developed in-house.

A prerequisite for the possibility to ensure the interface compatibility is that the interface descriptions are complete. The design of the interfaces is important for the design of the components, but may also affect the design of the verification and validation environments. The interfaces need also to be managed throughout the project. Note that this is valid also for interfaces with the environment that the product is operating in.

The actual assembly of components should be done in accordance with the selected integration sequence. However, before a component is included in the product, the readiness for integration should be confirmed. The identity of the component needs to be established and the conformance to the specifications and established criteria should be confirmed. This confirmation can include a check of the status of the component, e.g. that the design of the component is reviewed, that the component is tested and that the interface descriptions are followed. Once assembled, the components should be evaluated. This is done based on the integration sequence and the verification specified. Based on the systems created in the product integration process, the system is verified and validated. When all product components have been integrated, the product should be delivered to the appropriate customer. This can be made in an iterative fashion, with part deliveries, internal deliveries and of course as a final delivery for production.

2.2. Developing systems with CBSE

When developing a system based on components, the focus is on the system requirements, the overall system functionality and the mapping these requirements to components. However, the implementation of individual components is not in the focus of the process. The components used in the solutions are thus considered to be developed or acquired independently of the development of the system.

The activities performed when developing a system are similar to those for any non-component-based development; they include requirement analysis, architectural specification, component selection and evaluation, system design, implementation, integration, verification and validation. A specific activity here is component selection, but also other activities have specific parts that are influenced by the component-based approach. As the dependencies between these activities are strong, it is important to note that they are usually performed in an iterative fashion, and that these iterations should be taken into account when planning the system development.

The requirement analysis is done to transform the collected needs into system requirements. The task is also to define the scope for the system. Based on the system requirements, it is possible to define the system architecture and to derive the component requirements. As the definition of components to be used and the resulting system properties are investigated, it may be necessary to reexamine the system requirements and prioritize what is most important. The reasons may, for example, be that requirements are found to be contradictory, that the selected solution is too expensive or that the time-to-market requirements cannot be met.

When an initial architecture has been created, a decision how to obtain the needed components is taken. If the decision is to develop a new component, specific for the system, the development will be based entirely on component requirements derived from the system requirements. This decision will also make sure that the component

fits to the architecture. Preexisting components developed in-house may be used as-is, but may also require modifications. As this reduces the possibilities for reuse, it is more likely that interactions between the components are modified, that adapters are created, or that the architecture is modified to fit the selected components. This is also likely when using commercial components, as these normally require a specific architecture. Both types of pre-existing components may influence the architecture, especially if a specific component framework is required. To find and select components based on the component requirements is a challenge. One reason is that it is difficult to derive these requirements from the system requirements. If the component is not created specifically for the developed system, it is unlikely that a component exactly matching the requirements can be found. In addition to fulfilling the requirements, the components must also coexist in the system, which leads to the need to investigate compatibility issues between the components and also with the selected component framework. It is worth to mention that already in the selection process, integration activities can be performed. Often when validating components they must be composed with other components and integrated in the system environment.

The system construction depends on the chosen architecture and on the selected component technology and framework. The design also depends on what types of components will be used in the system. More reuse and commercial components will reduce the freedom to select different design solutions.

The implementation activities should be limited to adaptations of the components and connections between the components. This should be a minor task, but if the components are not properly selected, the work may be substantial. Also verification of the component behavior in the selected environment should be a part of the implementation. This may lead to additional development of code to handle the components in- and outputs or changes in the way the component is set up.

To ensure that the quality requirements on the system can be met, the integration of the system is crucial and should be started as soon as possible in the development cycle. The activities include determination of integration sequence, verification that the components adhere to the interface description, and provision of systems appropriate for verification and validation. Additional tasks are to identify the need for additional implementation and to monitor the system properties as these emerge when the system is integrated. The integration will depend on the architectural solution, as the possibility to build systems is determined by the selected architecture as well as the component model and framework. The verification that the requirements are met can start as soon as the first integration has been made, while the validation that the customer expectations are met can only be made when the final assembly has been made.

In component-based software systems, components may exist also in runtime. The result of this is that it is possible to change the system while in operation, or at least

without replacing the entire system, by replacing components. This simplifies the maintenance and error correction and also makes enhancements possible. A well-designed architecture is however necessary as the dependencies between different parts and components in the system make such changes dangerous if the consequences are not well understood. Special care must be taken when a component is used by several other components.

There are many reasons why component-based approach can improve the integration process. We list here the most important.

- Component specification. The basic principle in component-based approach is a separation of component specification from its implementation through its interface. This separation is stronger than in object-oriented approach since all interaction is supposed to be performed through interfaces. This principle drastically decreases the risks for introduction of unknown properties and architectural mismatches. Though it should be noted that many component models do not follow this principle, in particular for required interface, which may cause many unpredictable problems.
- Early integration requirements. For component validation usually a kind of integration procedure must be made. An early integration process can show problems that might remain hidden until the final integration.
- Standardized interoperation. Component models define the standards for interconnection between the components. This eliminates a number of potential errors due to architectural mismatches.
- Integration tool support. Integration is an inherent part of a basic approach of CBSE. For this reason the component-based technologies focus on this process and usually provide powerful integration tools.

2.3. Related work

This section describes some of the work that has been done related to integration in component based software systems. In the related work, the integration process partly includes what is often described as the composition process.

The notion that all development phases, including the integration activities, need to be reconsidered when working with component-based software is pointed out in [4]. It is also mentioned that the current component models do not take enough of the needs of the system developer into account. A part of the information that is mentioned as underdeveloped is the specific collaboration rules for interfaces and component behavior. This influences the ease with which a developer can determine if the chosen components fulfill the requirements of the system.

The PECOS project [5] [6] describes an approach and a software process to be used for basing embedded systems on component-based technology. The composition process is examined and described. It is, however, not compared to the overall expectations on the integration process.

The OOSPICE project [7] was targeted at overcoming the shortcomings experienced when applying software process improvement approaches to component-based development. In [8], the observation that component-based development is integration-centric is elaborated.

In [9], the risks in the composition phase for component-based software development are listed. Several of the risks are related to the integration process, and a method for how to deal with these risks is outlined.

3. Case study

The case study was performed at an ABB unit developing industrial control systems. The system has evolved through several generations, and a new generation of the system is currently being developed. Compared to the first generation, where the effort was three man months, the effort for software development in the current development is estimated to about 100 man years.

In essence, the controller has layered architecture and within layers, component-based design. The implementation consists of approximately 2500 KLOC of C language source code divided in 400-500 components, organized in 8 technical domains. The software platform defines infrastructure that provides basic services like: a broker for message-based inter-task communication, configuration support, persistent storage handling and system startup and shutdown.

3.1. Research method for the case study

The methods for the case study include interviews, document reviews and an observation. The interviews have been based on a set of open questions, and have been conducted as discussions about the integration process. The document review was performed on the documentation describing the integration process, the training material for the organization as well as the files used for and as a result from the build process. As the purpose of the observation was to identify challenges, it was designed to obtain as much information as possible, i.e. the decision was to perform an unstructured observation.

3.2. Product Integration

The development of the system is conducted in different development groups, and there are separate groups for the integration, verification and validation activities. As the system has evolved over several years and parts of it have been replaced with new solutions, the development environment as also been changed. For example two different configuration management systems are used. Unique tools are used for the integration group that also handles the build process. Developers have their own set of tools for building on local systems. Training of the developers is done as part of the general information about the system given to the staff. The developers also get hands-on training in the projects.

The system evolution is performed in an incremental way. The implementation of a functionality described in the requirement specification is distributed to different integration points (IP), as shown in figure 1.

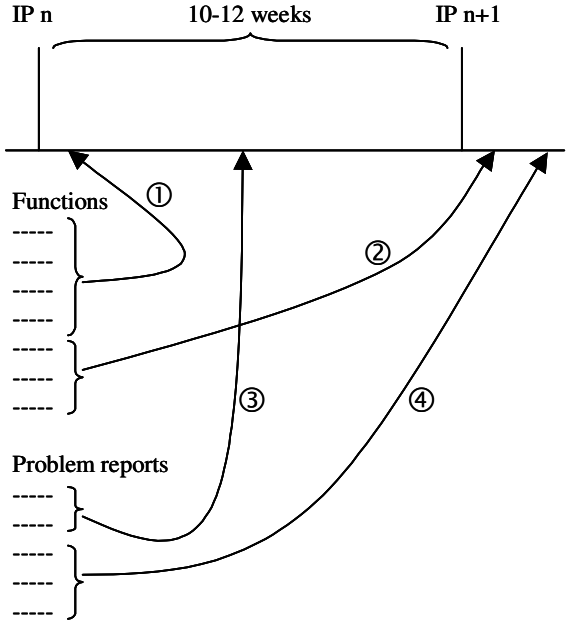

Fig 1. Distribution of functions and error corrections

The changes may occur in a project where the intended functionality for IPn is redistributed to IPn (1) and to IPn+1 (2). This redistribution is based on the progress in the project, the priorities for the different functions as determined by product management and the possibilities to alter the decided integration strategy. Also the problem reports and the error corrections related to them are assigned to the different integration points (3 and 4). Product and technology management decides what errors should be corrected for a specific integration point.

The procedure used when reaching an integration point is shown in figure 2. The width of the arrows in the figure (4) represents the amount of new functions or error corrections that are accepted for integration. As an integration point is approached, the possibility to add new functionality is reduced and increasingly monitored. This is illustrated by the narrowing towards the point of the arrow (1). As the "beta drop" is reached, the version is branched to a release track. All release tracks are made available to the organization for use in testing and further development. Errors that are found in the verification and validation are considered for correction for the new integration point (2). After the release "beta drop", the development groups have the possibility to add new functionality again (3).

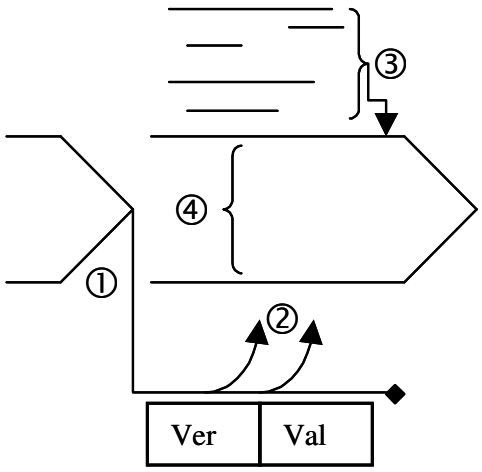

Fig 2. Integration point activities

An important prerequisite for a working product integration process is an appropriate build process. It is also in the build process that many of the problems with the product integration process appear. For our case study system, the current build process has been in place for four years and is continuously updated and improved. Each day, the full system is built and generated for several target systems with a total of more than 15 versions. A separate build machine is used, and each build takes seven hours. As soon as a build is started, it is possible to start delivering to the next one. New code to be included in a system build is put on a build queue. Once put in the queue, the component cannot be deleted from the queue. The two different software configuration management (SCM) systems used give different protection against mistakes. One prevents mistakes, as there are no possibilities to check code directly into the build directories. The other SCM system makes a direct merge into the release directory without the delivery through the queue.

The build is normally done during night, so the result of the build is known in the morning. The person responsible for execution of the build process examines the log files. In case of problems, the responsible persons are notified and asked to correct the problem. The result of a severe problem is normally that the build will be delayed one day. However, as the deliveries in the new build queue can be included, the setback may be different for different parts of the project. Today, no metrics or statistics are captured how often the problems occur or to see what causes the problems in the integration process. The error reports from the findings are however tagged with the build identity to make error correction easier.

The problems identified in the case study relate to three main areas. The first issue is the delivery of code to the build process. The code may be delivered late, or a function is not fully delivered. Also, the two different ways to deliver the code for integration is a concern. One system handles this automatically, while the other requires manual checking that the right things are included. The second issue is the

low quality, e.g. errors that cause the builds or initial integration tests ("smoke tests") to go wrong. This can be due to insufficient tests and system generation by the developers. They normally test only a few of the possible combinations. The result may be that the system generated works for the tested configurations but fails in the others. The final issue relates to components that influence other parts of the system. It may be that changes in include-files affect other components. This is possible as no routine or mechanism for how to handle the communication of changes has been established. This and the second issue may be discovered in the smoke test following the system generation.

4. Analysis

When we compare the problems discovered in the case study to the product integration expectations as described in [3], we see several activities that can be put in place to improve the process. The improvements of course can be made without the introduction of CBSE. However, our analysis of three main problem areas supports the idea that a CBSE solution would reduce the difficulties.

A first improvement is related to the checks at integration time and deals with the first two problems, delivery of incomplete functions and code with low quality. The rules for including a component at an integration point should be appropriate so that they can be followed both for major additions of functionality and for minor error corrections. This means that the rules should be suitable for different types of changes, but need to be followed for all inclusions at an integration point. To enable this, additional power must be given to the integration team. The development groups will through this lose some control but in return less often get unstable systems or broken builds. The improved check at integration time would be supported by CBSE as the delivery of code to integration would be done as ready-made components. This would also reduce the problem of functions delivered before they are ready. Through the use of CBSE, the poor quality can be reduced, as components should be tested in all environments they are envisioned to be used in.

The third and maybe most important problem area is the need to handle dependencies, i.e. interfaces, between different components more strictly. Changes to interfaces should be controlled and communicated. To achieve this, the interfaces must be sufficiently documented. Also, any changes to the interfaces must be controlled at integration time to ensure that they have been approved and communicated. In CBSE, the separation of the processes for developing components and for building systems into two separate processes helps in better defining the interfaces for the components. A component without a clearly defined interface cannot be used unless the developers of the system have full knowledge about the component. Introducing a clear separation in this manner would also increase the clarity in the dependencies between the components. It would also make it possible to have a more thorough, or strict, procedure for accepting a new version of a

component for a specific integration point. Using CBSE, improved descriptions of interfaces would diminish the influence from one component to another, or at least make these dependencies visible.

For all three main problems, we predict that CBSE would help in reducing the problems. The cost is however that the system, processes and organization need to be changed to accommodate CBSE.

A first step would be the introduction of a complete component model. There are experiences that by introduction of component models have significantly improved the development process [2]. Of course introduction of a component model would require additional efforts. First the existing code and basic architecture should be reused as much as possible. This implies that widely used components models such as .NET or EJB are not appropriate. Rather a simple, probably in-house developed component model should be deployed. This component model could be built incrementally, starting with basic principles such as interface specification and automation of integration of components.

A second effort required would be a componentization of the existing code. Since today many of the dependencies between the components are implicit, their separation might be a tedious work. However such a work would pay off in the long run, since errors made today depending on hidden connections between components would be reduced. Efforts to describe the dependencies explicitly are being made in the case study system today, with promising results. A continued work in this direction would result in an architecture that is properly documented and better cohesiveness of components which are the basic prerequisites for efficient system development and evolution.

Finally, the organization of projects and departments to clearly divide the work into development of components and development of the system is needed.

5. Conclusions and future work

A case study has been compared to the generic requirements on a best practice product integration process [3]. In addition to this, we have analyzed what support the current process may get from using component-based software engineering. Our conclusion is that several of the requirements for a well working integration process can get substantial support through skilled use of well defined components. The support comes from the fact that components should be well documented, tested in the environment they are intended for and that any dependencies to other components (or the environment) should be explicitly highlighted.

Future work should include additional case studies in industry. Both development units working with components and with traditional software need to be further examined. These investigations need to include measurements on the problems caused by an insufficient integration process as well as root cause analysis. The

purpose of these investigations would be to confirm or refute the conclusions in this paper that CBSE helps in providing a platform for efficient and effective software product integration.

Further additional analysis should be done on a feasibility of full componentization of the systems. The efforts and return-on-investments for re-architecting and for development and introduction of a component model should be estimated.

6. References

[1] Szyperski, C., *Component Software -- Beyond Object-Oriented Programming*, Addison-Wesley, Reading, MA, 1998.

[2] Crnkovic, I., and M Larsson, *Building reliable component-based software systems*, Artech House, Boston, 2002.

[3] Chrissis, M.B., M. Konrad, S. Shrum, *CMMI*, Addison-Wesley, Boston, MA, 2003.

[4] Zeidler, C., "Componentware Glory and Crux for early industrial adopters", Object Oriented Programming conference OOP 2000, Munich, Germany, 2000.

[5] Winter, M., C. Zeidler and C. Stich, "The PECOS Software Process", Workshop on Components-based Software Development Processes, ICSR 7, Austin, TX USA, 2002.

[6] Müller, P., C. Zeidler, C. Stich and A. Stelter, "PECOS — Pervasive Component Systems", Workshop on "Open Source Technologie in der Automatisierungstechnik", GMA Kongress, Baden-Baden, Germany, 2001.

[7] The OOSPICE project, http://www.oospice.com. (Link valid April 2005.)

[8] Stallinger, F., B. Henderson-Sellers and J. Torgensson, "The OOSPICE Assessment Component: Cusomizing Process Assessment to CBD", in Business Component-Based Software Engineering, edited by F. Barbier, Kluwer Academic Publishers, Boston, USA, 2002.

[9] Kotonoya, G., A. Rashid, "A strategy for Managing Risk in Component-based Software Development", Euromicro 2001 CBSE workshop, Warsaw, Poland, 2001.

PAPER D:

CASE STUDY:

SOFTWARE PRODUCT INTEGRATION PRACTICES

Stig Larsson, Ivica Crnkovic
In PROFES 2005 Conference, Oulu, Finland, June 2005

Abstract.

Organizations often encounter problems in the Product Integration process. The difficulties include finding errors at integration related to mismatch between the different components and problems in other parts of the system than the one that was changed. The question is if these problems can be decreased if the awareness of the integration process is increased in other activities. To get better understanding of this problem we have analyzed the integration process in two product development organizations. One of the organizations has two different groups with slightly different integration routines while the other is basing the development on well defined components. The obstacles found in product integration are highlighted and related to best practices as described in the interim standard EIA-731.1. Our conclusion from this study is that the current descriptions for best practices in product integration are available in standards and models, but are insufficiently used and can be supported by technology to be accepted and utilized by the product developers.

1. Introduction

Through investigations of many development organizations developing products with software as an important part, we have seen that the product integration is one of the processes where many of the problems in product development become visible. The origin of the problems is often in other processes performed early in the development cycle. These problems can be reduced through an increased understanding of the needs from an integration standpoint. Today, not enough care is taken to ensure that the system requirements are considered when components and parts developed. Proper preparation, understanding and performance of the product integration are believed to resolve part of this problem.

Integration of products that include software is described in several standards and collections of best practices. These best practices are collected from different companies and organization and include areas that are considered to be of good use for the development organizations in different application areas. There is however a lack of independent research which shows whether the practices described in these

collections give the intended result when implemented in different organizations; a systematic validation of the practices is needed.

There are different perspectives from which the use of descriptions found in standards and models can be investigated and different questions to be answered. The first question is how it can be determined that the processes described in the standards and models are suitable for different types of development and the use of different life cycle models; are the generic principles of the descriptions valid for all types of product development? Another question is if an organization may run into problems even if the principles and descriptions are followed in a proper way. Are there ways to fulfill the principles described but not achieve the intended results? A third question is how to determine if the reason for an organization having problems is the fact that the principles are described as the prescribed working method, but are still not followed. Our approach to these different perspectives is to look at the performance of the process in the investigated organizations and compare the activities with the ones prescribed in the standards and models regardless of the development model used. We also look at the problems in the organizations and analyze these with respect to the practices that are not followed by the organization.

We claim that we by investigating a number of organizations and the practices in use can obtain support for the practices described in standards and models or determine a need for revisions of the standards and models. This leads to the following research questions for this paper: (i) How well can the practices described in a specific standard be expected to reduce problems encountered in the integration of products? and (ii) What deficiencies or incompleteness can we observe in the proposed practice?

We have in this paper selected to use the interim standard EIA-731.1 [1] as the reference model. The rational for this is that the interim standard model has been used as one of the inputs to CMMI [2], and is specifically intended to be used for internal process improvement, not for qualification of suppliers. In addition to this, the development of this interim standard has been carried out in cooperation between a number of national and international organizations such as EIA[3] and INCOSE [4] involving a large number of organizations and companies with substantial experience in software and system product development.

Our proposition in this paper is that the problems encountered in the investigated units relate to the lack of execution of practices that are described in the interim standard. We also propose that successful execution of the product integration can be mapped to specific implementation of practices described in the interim standard.

This case study is a continuation of the work described in [5], where a different case has been compared to CMMI. The purpose of this paper is to investigate one additional source for best practices, compare it to current industrial problems and to

establish if there are connections between the problems and the lack of execution of proposed activities.

The remainder of the paper is organized as follows. Section two describes general structure of the interim standard EIA-731.1 as well as the main characteristics of the integration processes of a development process. In section three, the case study design is described with explanations about the data collection method, the analysis method and the threats of validity of the study. Section four includes a description of the findings from the case study. Section five analyzes how the findings relate to best practices. Finally section six contains the conclusion and proposed future work and is followed by the references list.

2. Product Integration in EIA-731.1

The interim standard EIA-731.1 describes a number of focus areas useful for organizations developing products and systems. The focus areas described are organized in three categories; technical, management and environment. For each focus area, a number of themes describe the suggested activities. All themes include a description, typical work products and specific practices for the focus area. For some of the focus areas there are comments that normally contain clarifications or suggested implementation details. In addition to the specific practices, there are a number of generic practices applicable for all specific practices with the different focus areas. The generic practices include tasks such as planning of the activities to perform the process, monitoring and checking that the activities performed are according to plan and the execution of corrective measures when these are identified and needed.

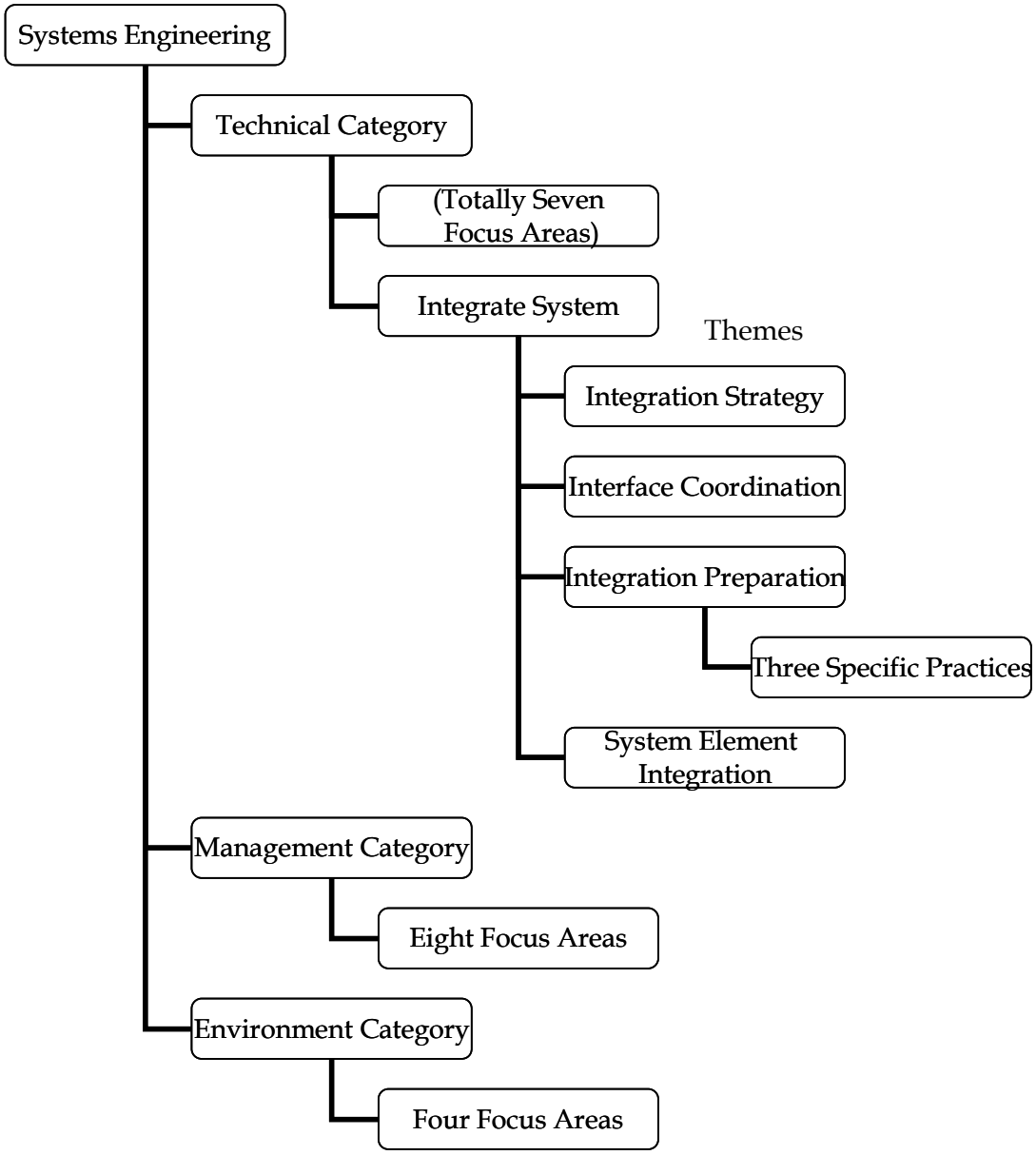

Fig. 1. Structure of EIA-731.1

The interim standard includes a possibility to determine the capability level of an organization in a specific area. This is based on the observation that organizations typically take observable distinct steps in the effort to improve the performance. In EIA-731.1 these levels are intended to be used as means to help the organization in the planning and implementation of the improvement efforts. Six different capability levels have been defined. Level 0 indicates that the specific practices are not performed. Level 1 indicates that the specific practices on level one are performed.

For level 2 to 5 both the specific and generic practices on these levels are performed. Note that no effort has in this study been made to determine the capability level of the organizations investigated as the target is to understand if the specific practices for product integration give the intended result.

The rest of this section summarizes the product integration process as it is described in EIA-731.1. The standard prescribes a set of specific practices that are considered to be essential for accomplishing the purpose of the focus area designated Integrate System (Focus Area 1.5).

The purpose of the Integrate System focus area is to ensure that the product and system works as a whole based on the components that have been integrated. Interfaces between components and functions that extend over many components in the system are in the center of attention. It is also noted that the integration activities should start early and are typically iteratively performed.

Four themes have been identified for the focus area. An Integration Strategy (1) is considered to be the basis for the integration process. This theme includes the development of a strategy that contains an integration sequence and a plan for the integration tests to be performed. The Interface Coordination (2) is the second theme and includes handling of the requirements on the interfaces as well as specifications and detailed descriptions. As a third theme, the Integration Preparation (3) describes how components are received for integration and the checking that the components are in accordance with the strategy and interface documentation. The final theme is the actual integration: System Element Integration (4). The components are integrated according to the plan and the inter-operations between the components are checked. It should be noted that the actual verification is described in a different focus area in the interim standard EIA-731.1, FA 1.6 – Verify System.

The different specific practices on capability level 1, 2 and 3 for all themes can be found in Table 4. The descriptions in the interim standard are short and need to be interpreted with the description of the theme as a basis. Some guidance can be found in EIA-731.2 [6] that describes an appraisal method for EAI-731.1. However, the sample questions in this guide are also on a high level and require substantial expertise to be used.

3. Case Study Design

The case study was performed on three different product development groups in two different organizations. As the development methods are different in all three groups, the case study has been designed as a multiple-case holistic study as described by Yin [7]. The units of analysis are the processes for integration as perceived by members of the development groups in the three different cases. The focus of the study was on processes used at the time for the investigation, not described in quality systems or handbooks and not on processes that were under development.

3.1 Research Method

The interviews made with members of the development groups are the main sources of data in this investigation. Additional information was obtained from descriptions and examples of how the integration was planned and performed. For each case at least two persons were interviewed. The selection of subjects for the interview was based on two criteria. The first was that for each organization, both a manager and a developer should be interviewed. The second criterion was that the subjects should have extensive experience spanning over several years from the development in the investigated group.

The interviews were performed as open-ended discussions and all interviews were made by the same researcher. The researcher was guided by a discussion guide to ensure that different aspects of product integration were covered in the discussion. The guide was developed by two researchers and included questions related to three different areas; organization, implementation, and effectiveness of the product integration. The questions included in the discussion guide were not taken from the standard, but were designed to give an understanding of the used processes independent from descriptions in standards and models. During the interviews, the guide was used to ensure that the interesting topics were covered, and the specific questions asked were depending on how much information was obtained through the explanations from the interviewees. The use of open-ended questions allowed the researcher to follow up interesting statements that lead to more information and a deeper understanding of the used process. Each interview was between one and two hours. The documentation from the data collection consists of notes taken during the interviews complemented with information from the written documentation.

The data collected can be divided into two types. The first type was descriptions of how the integration process was performed for each case and what activities were carried out. The second was descriptions of the problems that the units perceived in the integration process.

3.2 Analysis Method

After the interview sessions, the data collected was analyzed in several ways. This was done as a separate activity and without the involvement of the development organizations. For each case in the case study, the activities captured during the data collection were compared and mapped to the practices described in EIA-731.1. The result from the mapping showed if the development in the different cases were performed in accordance with the interim standard. As a second step, the problems identified were mapped to the specific practices in EIA-731.1 that are intended to ensure that the problems should not occur. Finally, the relations between activities performed and the problems were investigated. This resulted in Table 4 that indicates the relation between practices from EIA-731.1, activities performed and identified problems. A second phase of the analysis was to propose how the practices

in EIA-731.1 should possibly solve the encountered problems. The results from this analysis in found in Table 5. The analysis was made by one researcher and reviewed by two other researchers.

3.3 Validity

Four types of validity threats are of interest for case studies [7]. In this section, we discuss these and the preventive measures to reduce them. Construct validity relates to the data collected and how this data represent the investigated phenomenon. Internal validity concerns the connection between the observed behavior and the proposed explanation for this behavior. The possibilities to generalize the results from a study are dealt with through looking at the external validity. Finally, the reliability covers the possibilities to reach the same conclusions if the study was repeated by another researcher.

The construct validity is dealt with through multiple sources for the data through more than one interview for each case. Additional interviews with other stakeholders as well as additional document investigations would have increased the construct validity. However, this would have required more intrusive investigations and would limit the availability to the organizations. The design of the discussion guide was based on available standards and methods and involved more than one researcher to ensure that the questions to be discussed were relevant. The researchers experience in software product development provided a basis for relevant discussions under the interview sessions.

The internal validity was secured in three ways. First, the connection between the behavior and the interim standard was done in several steps to avoid predetermined connections. Secondly, rival explanations have been listed and examined to exclude other causes to the findings. Finally, the analysis of the data and the connection to the interim standard has been reviewed by two additional researchers to avoid personal bias.

The external validity is dealt with through the use and description of three cases in two different application domains and through the use of several different standards and methods when defining the investigation area.

The reliability of the study has been secured through the description of the procedure used in the study and the documentation of the discussion guide.

4. Case Descriptions

Two product development organizations have been investigated, both developing systems for monitoring and control of different types of networks, but in different application domains. The systems operate in industrial settings with real-time requirements as well as high demands on availability and reliability. One of the units is developing products for two different environments. This has lead to the use of

different processes and in this study they are treated as two cases resulting in a total of three cases. For each case the following sections contain a brief description of the product and the product development process. The descriptions also include the problems that were identified and described in the interviews. The problems are presented in tables where each problem is labeled with a P, the case number and a reference character.

4.1 Case One

The product in case one is a stand-alone product that is connected to a real-time data collection system. The development is done in one group with less than 20 developers and follows a clearly defined process. The product development of a specific release is based on a definition of the product that contains what should be included in each release. The first step in the development is the implementation of requirements on the functions for the release. Based on this, the unit and system verifications to be performed are defined. Development of the functions is done in units called components. The Rational Unified Process is used, and a document list defines the development process. The planning is made so the development is done in increments. The unit verification is performed by software developers. The strategy is that tests should not be done by the developer producing the software. The unit tests are often done through automatic testing. Specifications and protocols from the tests are reviewed by peers and system integrators. The tests are performed in the developer's environment and consist of basic tests. Functional tests are performed before the system tests.

The product integration is not defined as a separate process, but the product is integrated by the developers before the system verification. Before a component is checked in, it should be included in a system build to ensure proper quality. Delivery to the system test is done of the whole system. The test protocols and error reports from the unit verifications are reviewed with the system integrator before the system test. The system tests are performed by a core of system testers and temporary additional personnel. This strategy builds on well defined and detailed tests. The tests are focusing on functions and performance and are performed on different hardware combinations. This includes different variants of the product and different versions of the operating system. The test period takes approximately 12 weeks, with new versions of the assembled components received to system test every week. Although the development builds on increments, no integration plan is used for the product. The integration plan used is one for the whole system where this product is included. Typical time for the development of a release is less than one year.

The three most serious problems were captured for case one as described in Table 1. The routines are mainly followed, but due to tight deadlines, shortcuts may be taken. Sometimes uncontrolled changes are introduced in the software. This is typically done when a part of the system is changed due to an existing error that is uncritical

and not planned to be corrected. Due to the dependencies in the system, new errors may appear in parts that have not been changed. Also other connections between components that are not explicit generate this problem.

Table 1. Problems captured for case one

Label	Problem description
P1-A	Functions are not always fully tested when delivered for integration. This leads to problems in the build process or in integration and system tests
P1-B	Errors are corrected that should not be. This results in new errors with higher influence on functionality and performance
P1-C	Errors appear in other components which have not been changed

4.2 Case Two

The second case is a product that includes software close to the hardware. The development group is small and follows a common development process. This process includes rules for what should be checked and tested before a component is integrated. The tests include running the application in simulators and target systems before the integration. A specification for what should be ready before start of functional and system test are available. The architect is responsible for implementation decisions. The target system includes a complex hardware solution with the application divided on two target systems. Typical time for the development of a release is 1.5 year. This includes the full development cycle from defining the requirements to system testing.

Most of the problems appear because of the incapability and version mismatch of the test system, the final product and the test and final hardware platform (Table 2). Efforts are now made to go towards incremental development, and to increase the formalism in the testing. The tests will be made in three stages with basic tests performed by the designer, functional tests performed by a specific functional tester and system tests with delivery protocol.

Table 2. Problem captured for case two

Label	Problem description
P2-A	Problems appear as a consequence that tests for the components are not run in the same environment as the test system. Different versions of hardware and test platform are used.

4.3 Case Three

The development organization in this case is responsible for the design of a user interface that acts as a client to a database server. The organization is small, around 15 developers.

The current architecture has been recently improved. The old version of the system suffered from problems with many common include files. Through global variables and similar solutions permitted by the selected technology, unintended side-effects made debugging and error correction tedious. Different attempts to reduce the problems within the available technology lead to the insight that a design that was built on isolation of interfaces should be beneficial. The solution was to start building a new system. Included in this decision was a strategy to design interfaces carefully and to use technologies that permitted isolated components to be used.

The system is built up of components that primarily implements different parts of the user interface. Each component handles the communication with the server. This design was used to allow the development of services that are independent and dedicated for each component. The component framework defines the required interface for each component and provides a number of services, such as capturing of key strokes. The technology used permits the developers to easily isolate problems and to minimize the uncontrolled interference and dependencies between the components.

The development is organized with frequent builds and continuous integration of new functions. The integration is handled by the integration responsible. However, the checks before the inclusion of new functions are done by the developers. There are no specific routines in place for handling the interfaces. Changes are in practice always checked by the system architect.

The new system design has reduced the implementation time for a function with 2/3. The turn-around time for a system release has been reduced from six months to between one and three months. At the same time, a need for maintaining the base platform has emerged. Also, some of the technical solutions have been questioned and may increase the need for maintenance (Table 3).

Table 3. Problem captured for case three

Label	Problem description
P3-A	Scattered architecture on the server side as a result of the decision to handle communication in each component

5. Collected Data and Analysis Results

In these three cases we found may similarities: size of the development groups, similar concerns, requirements of the products, similar product life cycle. What we have seen are the differences in the development processes and in used technologies and approaches. Our intention is to analyze what are the sources of the main problems and if they could have cause in deviation or absence of the activities pointed out in the best practices.

This section contains two parts. The first includes a table containing the analyzed data from the case study, while the second lists the problems found in the cases with a suggested implementation of the practices that could improve the performance.

5.1 Analyzed Case Study Data

The three steps of the analysis have been summarized and presented in Table 4. The table includes two parts for each practice. The first two columns show the description from EIA-731.1 for the specific practices for the focus area Integrate System. The first number in column one shows what theme the practice belongs to, and the second number is the capability level (i.e., 1-2 shows that the practice belongs to theme one and is placed on capability level 2). Finally, if two or more practices exist on a capability level for a theme, these are distinguished by a character. The following three columns include data from each of the cases. These columns include two things: (i) an indication for each case if the practice has been observed as performed (+) or not observed (-), and (ii) if there are indications of problems connected to the practice (*). The indicated problems are further described and analyzed in section 5.2.

5.2 Analysis of Observed Problems

In each of the cases, problems encountered in the performed product integration process were captured and discussed. The problems are in Table 5 cross-referenced by the researcher to the specific practices for the Integrate System focus area of EIA-731.1. Each problem has a label composed of a P, the case number and a reference character as in the tables in section 4. In addition to the description and the reference, a proposed action based on the specific practice has been included in the table.

Based on the data, we have made two observations regarding the perceived problem situation. The first is that all the problems for case one and two are related to capability level 1 specific practices. This may indicate that additional problems may be observed once all capability level one practices are performed, or it may indicate that higher capability level practices have less influence on the actual product integration results. The second observation is that case three had a similar culture for process adherence as case one, but the developers were forced by the technology to perform the specific practices.

Table 4. Specific practices for Integrate System compared to data from case 1, 2 and 3

Specific Practice	Description	Case 1	Case 2	Case 3
1-1	Develop an integration strategy	+ *	+	+
1-2	Document the integration strategy as part of an integration plan	-	+	-
1-3a	Develop the integration plan early in the program	-	+	-
1-3b	When multiple teams are involved with system development, establish and follow a formal procedure for coordinating integration activities	-	-	-
2-1a	Coordinate interface definition, design, and changes between affected groups and individuals throughout the life cycle	- *	-	+
2-1b	Identify interface requirement baselines	- *	+	+
2-2a	Review interface data	-	-	-
2-2b	Ensure complete coverage of all interfaces	-	-	-
2-3a	Capture all interface designs in a common interface control format	-	-	-
2-3b	Capture interface design rationale	-	-	- *
2-3c	Store interface data in a commonly accessible repository	-	-	-
3-1a	Verify the receipt of each system element (component) required to assemble the system in accordance with the physical architecture	- *	- *	+
3-1b	Verify that the system element interfaces comply with the interface documentation prior to assembly	- *	+	+
3-2	Coordinate the receipt of system elements for system integration according to the planned integration strategy	-	+	-
4-1a	Assemble aggregates of system elements in accordance with the integration plan	+	+	+
4-1b	Checkout assembled aggregates of system elements	+	+	+

Table 5. Cross-reference between observed problems and relevant specific practices

Label	Problem description	Relevant specific practices and proposed actions
P1-A	Functions are not always fully tested when delivered for integration. This leads to problems in the build process or in integration and system tests	3-1a Ensure a handover to a dedicated integration responsible
P1-B	Errors are corrected that should not be. This results that new errors are introduced, with higher influence on functionality and performance	1-1 Ensure that the strategy and decision are followed through a handover procedure
P1-C	Errors appear in other components than the changed	2-1a, 2-1b, 3-1b Specify and enforce interface descriptions for all dependencies between the components
P2-A	Problems appear as a consequence that tests for the components are not run in the same environment as the test system. Different versions of hardware and test platform are used.	3-1a Ensure that the proper test equipment as described in the integration strategy is made available to the developers. Check that proper tests are performed through a clear handover to an integration responsible
P3-A	Scattered architecture on the server side as a result of the decision to handle communication in each component	2-3b Ensure that the rationale for design decisions are documented and communicated

5.3 Analysis of Propositions

As a summary of the analysis, we conclude that case two is performing the product integration most in line with the specific practices described in EIA-731.1 It is also clear that case two and three follow almost all the recommendations from capability level 1 specific practices. We see that case one has the most problems, and that all these problems are related to capability level 1 specific practices and we have noticed that in case three, the technology may help the development team in following the capability level 1 practices. The results are displayed in Table 6.

Table 6. Summary of analysis

	# of specific practices performed of total number # of problems found		
	Capability level 1	Capability level 2	Capability level 3
Case 1	3 /7 5 problems	0/4 No problem	0/5 No problem
Case 2	5/7 1 problem	2/4 No problem	15 No problem
Case 3	7/7 No problem	0/4 No problem	0/5 1 problem

The first of our two propositions was that the problems encountered in the investigated units relate to the lack of execution of practices that are described in the interim standard EIA-731.1. In the analysis of the data and the comparison, we conclude that the problems found can be mapped to specific practices which support our proposition. We have also observed that it is primarily the inability to perform capability level 1 specific practices that have lead to observable problems.

The second proposition was that successful execution of the product integration can be mapped to specific implementation of practices described in the interim standard. For many of the practices on capability level 2 and 3, no observations have been made that they were performed, but only one problem has been reported that could be related to level 2 or 3 practices. Based on this and the observations regarding capability level 1 practices, an additional proposition has evolved and should be tested in future studies. This can be formulated as follows: A successful execution of the product integration can be mapped to specific implementation of practices described in the interim standard for capability level 1.

5.4 Rival Explanations

The conclusion regarding the propositions above can be challenged and in this section we examine rival explanations and analyze the possibility that these give better reasons to the data found in the study.

The first explanation examined is that there is no real connection between the performance and the specific practices described and that the data match only is coincidental. We consider this explanation to be unlikely due to two facts. The first is that the interim standard build on long industrial experience from companies and organizations from a wide set of areas and applications. The second fact is that the pattern shown in this study is clear and builds on three cases from two different organizations.

The second alternative explanation could be that the organizations due to other factors succeed in the product integration process. However, if there are other factors

involved, these may also help in following the proposed practices. This is also the situation in case three where the selected technology has imposed a way of working on the product developers.

6. Conclusions and Future Work

Data regarding the product integration process from two development organizations have been collected and compared to the requirements described in a standard description of the product integration process. The problems observed in the case study have been compared to practices that describe activities that should improve the performance in the product integration.

We can from the observations conclude that the basic level of practices described in the interim standard EIA-731.1 includes activities that can help the organizations to avoid problems which can appear when integrating components to systems. Basic activities include (i) development and a clear specification of the strategy for the integration, (ii) keeping well defined interface descriptions up to date throughout the life cycle, (iii) that the integration of components follow the strategy and (iv) that the assembly is verified as planned.

We have also observed that there are indications that skilled use of component technologies as described in [8] facilitates the integration process. The factors contributing to this support are well described interfaces, the need to test components before integration and the explicit definition of the environment required by the components.

Through this investigation, partial answers have been found to our research questions, but additional research is needed. Future work should include steps to strengthen and further investigate the propositions made in this paper. They are (i) improvement of validation of the results by providing the feedback to the case participants in a form of discussions of accuracy of collected data and the results at a common workshop, and (ii) additional case studies in industry. Additional descriptions of practices in standards and models need to be investigated in relation to industry practices. There is also a need to analyze the similarities and differences in the different standards and models. One additional research direction has been indicated with the purpose to confirm or refute the indications in this paper and in [5] that component technologies assist in the implementation of successful software product integration. Of specific interest may the integration problems related to COTS be.

References

[1] EIA/IS-731.1, Systems Engineering Capability Model, Electronic Industries Alliance (Interim Standard), (01 Aug 2002)

[2] Chrissis, M.B., M. Konrad, S. Shrum, CMMI, Addison-Wesley, Boston, MA, (2003).

[3] http://www.eia.org/. (Link valid April 2005.)

[4] http://www.incose.org/. (Link valid April 2005.)

[5] Larsson, S., I. Crnkovic, F. Ekdahl, "On the Expected Synergies between Component Based Software Engineering and Best Practices in Product Integration", Euromicro Conference, France, August 2004, IEEE

[6] EIA/IS 731.2, Systems Engineering Capability Model Appraisal Method, Electronic Industries Alliance (Interim Standard), (01 Aug 2002)

[7] Yin R. K., *Case Study Research: Design and Methods* (3rd edition), ISBN 0-7619-2553-8, Sage Publications, 2003

[8] Szyperski, C. et al, *Component Software -- Beyond Object-Oriented Programming*, (2nd edition), ISBN 0-201-74572-0, ACM Press, New York, (2002)

PAPER E:

EXPECTED INFLUENCE OF ETHICS ON PRODUCT DEVELOPMENT PROCESSES

Stig Larsson
In ECAP Conference, Västerås, Sweden, June 2005

Abstract

Product development efficiency and effectiveness is depending on a process being well executed. The actions of individuals included in the processes are influenced by the ethical and moral orientations that have been selected by each individual, whether this selection is conscious or not. This paper describes different ethical choices and the expected effects they may have on the development process exemplified by the product integration process for software products. The different frameworks analyzed are utilitarianism, rights ethics, duty ethics, virtue ethics and ethical egoism. The expected effects on the goals for product integration may be debated. This is a result in it self as it triggers discussions about ethical considerations and increase the awareness of the influence of moral decisions. Our conclusion is that the adherence to specific moral frameworks simplifies the alignment of actions to the practices described in product development models and standards and through this supports a more successful execution of product development projects. This conclusion is also confirmed through a comparison between the different directions and several codes of ethics for engineers issued by organizations such as IEEE as these combine features from several of the discussed ethical directions.

1. Introduction

The application of different ethical approaches in product development organizations is likely to influence the effectiveness and efficiency of product development [1]. This is based on the assumption that actions performed by individuals involved in product development depend on the moral values that generally govern all areas of life. Ethical considerations can be investigated from different viewpoints; organizational, management, group and individual. The analysis here is concentrated on the choices made by the individual developer. We suggest that a conscious decision by the individual to base actions on a specified set of ethical rules shape how successful interactions with co-workers will be, how well different tasks are performed and eventually how professional the development is executed.

Throughout history different ethical theories have been formulated and expressed. To analyze these, a categorization is needed and in this paper we follow the classification made in [1]. Five different moral frameworks have been selected and for each of those one or two different versions are described and analyzed from a product development perspective. The five frameworks are utilitarianism, rights ethics, duty ethics, virtue ethics and ethical egoism.

Numerous standards and reference models are available defining the processes needed to develop a product [2][3][4][5]. We have selected to investigate the Product Integration Process and concentrate on the case where the product is primarily based on software. This selection has been made as it highlights communication between different engineering disciplines and it relies on trust between co-workers.

The rest of the paper is organized as follows. Section two describes a number of ethical directions. Section three introduces the Product Integration Process which is used as an example of a part of the product development process. In section four, the different ethical directions are applied to the actions by individual software developers in the product integration process, and the consequences are discussed. Section five contains a comparison between the different moral orientations with the IEEE Code of Conduct [6]. Section six contains a conclusion as well as proposed future work in this area.

2. Ethical directions

Ethical theories describe and give a generalized view of moral issues, putting the concerns into perspective. Ethical concepts are used in different ways. Descriptive ethics try to describe values and moral without deciding if an action based on these is right or wrong. Normative ethics go one step further and give more guidance in moral questions and choices as this approach contains questions regarding the duties and values. Applied ethics examine the moral choices that are made in specific situations and areas of interest. One such area is Engineering Ethics which covers the professional considerations for engineers and product developers. Several Codes of Conducts are available expressing different professional organizations' opinions about and commitment to ethics [6] [7].

2.1 Utilitarianism

The basic idea of utilitarianism is that where there is a choice to be made, the action that is best is the one that brings the greatest happiness for the greatest number of people. This means that for each decision, there should be a possibility to calculate the optimal way to go. It should be noted that the extent for the calculation is the whole society, which differentiates utilitarianism from pure cost-benefit analysis which normally has a much narrower scope. In order to be useful, utilitarianism requires both a set of values that specifies how to measure happiness and the ability to predict what actions would secure that happiness. These predictions are normally

provisional and need to be revised when more facts are known and may also lead to changes in decisions, if possible. The two versions described here are act-utilitarianism and rule-utilitarianism differ with regards to the predictions.

The focus for act-utilitarianism is the deeds in each situation. This means that each specific action is right if it brings the most good for the most people. Both the long-term and immediate effects should be taken into account, and alternatives considered when the consequences are predicted and calculated.

Rule-utilitarianism instead focuses on a set of rules that together can bring the best to most people. This means that the goal would be a set of rules, or a moral code, for the society that if used by all people would maximize the public good. This version is thus more indirect and requires actions to be compared to the selected set of rules, instead of predicting the consequence. This idea is also one of the basis for different codes of ethics for engineers.

2.2 Rights ethics

Fundamental to rights ethics is the respect for the individuals' dignity and value. This is contrasted with the good of the society emphasized in utilitarianism. Two distinct versions exist, liberty rights with emphasis on the right for the individual to have the freedom to perform actions without interference from others, and welfare rights that concentrate on the right to have the possibility to live a decent life for everyone regardless of capabilities. These complement each other and most rights ethicists agree that both types exist. The rights described are depending on the context, and are also connections to the legal rights in the society. This distinction is however clear. Legal rights are expressed in the laws for each society, while the human rights are considered to exist even if they are not reflected in the laws.

The liberty rights are based on the notion that we should respect each individual and that person's dignity and value as she or he performs actions demonstrating the liberty. In a professional context, this means that the actions that we take in our work should not be meddled with as long as we are within the confines of our jurisdiction, and other people should respect our choices.

Welfare rights depend on the societal context as the possibility to require the community to assist individual depend on the availability in the community. Transferred to a product development context, this could include the support for individuals and project parts that need assistance to be able to fulfill their tasks, but could also be governing the right to limited working hours without considering the need to fulfill organizational needs.

2.3 Duty ethics

Duty ethics is connected to the rights ethics, as they complement each other. The duty for one person can be the right for another person. One example from

engineering could be the duty to deliver something when promised with the corresponding right to receive it when it is needed.

Immanuel Kant (1724-1804) based the discussion of what duties we have on the fundamental duty we have to respect persons. This gives autonomy to each individual, but also the duty for each individual to make choices that would be acceptable if anybody made that choice.

2.4 Virtue Ethics

The emphasis in virtue ethics is on character. This means that rules and rights are secondary and a result of different desirable features that have been identified. These include competence, fairness, honesty, and loyalty. In the professional life, the virtues may be directed towards different scopes. The requirements and virtues for public, teams, the profession, and self-governance differ and complement each other.

The set of virtues may be easy to decide on, but the judgment of individual actions if they are virtuous is not easy. Very often, there is a thin line between a virtue and a vice, like courage and fool-headedness.

Different ethicists emphasize different virtues. Two examples are Florman [8], putting special emphasis on loyalty to employers, and MacIntyre [9] that stresses the loyalty to the community.

2.5 Ethical Egoism

The basic idea of ethical egoism is that each individual should endorse self-interest and maximize the personal well-being. This should not be short-sighted, but look for a long term situation where the social contract is still honored. This includes following the laws and other societal agreements that in the long run would benefit the individual. The result of egoism is thus that the caring of others and compassion is not of value, and should not be the basis for actions.

One variant of ethical egoism includes a more community-oriented approach. The self-realization is in center, but should be complemented with the importance of humans as social beings and the need for communities and relationships with others to ensure the individual well-being.

3. Example process: Product Integration

The endeavor of developing a product can be described as the execution of a set of processes. Different standards and reference models describe the requirements on these processes [2][3][4][5]. Several of these are collections of experiences that form what is considered to be best practices for the difference processes. In this paper, the example process is the Product Integration Process. Product integration involves several different groups of engineers and efficient communication as well as a high degree of trust is crucial for successful execution.

The Product Integration Process for software products represents the activities to combine software components to a product and to ensure that this product has the expected functions and qualities. The goal is to deliver a product that fulfills the expectations of the customer. It is expected that projects that have a Product Integration Process that follows the practices described in different reference models will have a higher probability to deliver on time with expected quality.

In this paper, we have selected a reference model from the Software Engineering Institute, the Capability Maturity Model Integration (CMMI) [2]. CMMI version 1.1 defines 25 process areas, and for each process area there are a number of practices that, if performed, represents an indication of maturity. It is expected that this also increases the performance of the development organization.

Three goals are defined for the Product Integration process area: (i) prepare for product integration, (ii) ensure interface compatibility and (iii) assemble product components and deliver the product.

The three specific practices for the first goal are connected to each other. The basis is the integration sequence strategy and the integration sequence that build on this strategy. Besides the product components, the test components and equipment need to be included in the integration as the project progresses. This leads to the second practice which is to establish the environment for the integration. The build-up of this environment needs to be included in the integration planning based on the decided integration sequence. The different engineering disciplines such as software development, integration and test need to have a close cooperation to ensure that the plans are realistic and that all needed equipment is included in the planning. This cooperation is also needed for the third practice which is to establish the procedures and criteria for product integration.

The second goal describes the need to ensure that the different parts fit together. This can be achieved through two practices. The first is the review that is needed to make certain that the descriptions of the interfaces are complete, while the second is the need to manage the interfaces throughout the project life-cycle.

The actual combination of the different parts of a product is described in the third goal and is supported by four practices. The first is a preparation that includes checking that the delivered components adhere to the criteria for integration that has been established. The second is the actual assembly activities. These activities should follow the selected strategy and integration sequence. After the compilation, the product should be evaluated with specific care in testing and evaluation of the interface interactions. The final activity is the packaging and delivery to the customer.

4. Applying Ethical Directions on Product Integration

In this section, different ethical theories are related to the goals and practices for Product Integration as described in the CMMI and given in table 1. The analysis is done from the view of a software engineer responsible for the development of a specific function in a software product. For each of the theories, the question if it supports each of the goals is considered. An indication is given for each of the goals if the theory can be expected to support or oppose the intentions with the goal. There are also indications if our analysis is inconclusive. The conclusions in this section can be debated, and this is probably the most important result as this triggers the discussion regarding ethical considerations in the development of software products.

4.1 Act-Utilitarianism

For each situation, actions should bring the most good for the most people, and both immediate and long term effects should be considered.

Prepare for Product Integration. The idea that we should maximize the good for all people does neither help nor oppose the preparation. As an example, if the developer synchronizes the integration sequences ad-hoc in a successful way, this benefits most people as the goal of the project is fulfilled, but does not fulfill the goal of the process. On the other hand, the existence of this goal in CMMI indicates that there are benefits for the project and resulting product in having a strategy for the integration sequence.

Ensure Interface Compatibility. Also this goal may be supported or not. In general, the number of errors found in later stages of product development will be reduced if the interface compatibility is ensured. On the other hand, in the short term, this leads to additional work for all involved engineers, which may already have ensured interface compatibility in the development of the function.

Assemble the Product Components and Deliver the Product. The goal to maximize benefit is normally supporting the goal to assemble the product and bring it to the market. Of course there are exceptions where products do harm to many persons, but generally act-utilitarianism supports this goal.

4.2 Rule-Utilitarianism

For each situation, a set of chosen rules that should bring the most good for the most people is to be applied. If the rules are carefully selected, all three goals should be supported. However, one alternative to a well working product integration selected by some organization is to test extensively before a product is released. This may be a way to maximize the benefits for most people, but does not ensure that the goals for product integration are fulfilled.

4.3 Liberty Rights Ethics

The freedom to act for each individual should be respected. For the engineer, this could mean that as long as the result of the work is leading to the common goal, the means to that goal is a free choice for the engineer.

Prepare for Product Integration. The preparation requires that a strategy for the sequence of integration is selected and implemented. This is supported implicitly by the liberty rights, as it does not prescribe how the engineer meets the requirement on delivery on a specific time. Also the build up of environment and the specification of rules is supported, as this makes it easier for the engineer to understand the constraints for the development of functionality.

Ensure Interface Compatibility. Ensuring interface compatibility requires review. This can be considered as an infringement on the freedom to act for the engineer and that the results delivered are not respected. The conclusion is that this goal is not supported by liberty rights ethics.

Assemble the Product Components and Deliver the Product. The assembly and delivery of the components and the product depend on the result of the engineering work. Of course, the quality of the product is important, but the procedures to achieve it are not prescribed for this goal. Hence, the engineer is free to do what is required within the constraints, and consequently the goal is supported.

4.4 Welfare rights ethics

Transferred to a product development context, welfare rights ethics can imply the support from the organization to the individuals that need assistance to be able to perform the task.

Using this interpretation leads to an inconclusive result regarding the impact on all three goals for product integration. The needs for the individual may increase the focus on achieving the goals. One effect of this could be that engineers needing assistance to perform their task would always get it, and this would lead to better fulfillment of the goals. On the other hand, it might lead to sub-optimization and divert the work from the organization's goals. The influence will hence depend on the possibility for support within the resource constraints given for different parts of the organization.

Table 1. Relation between Product Integration goals and ethical directions

	Goal 1: Prepare for product integration	Goal 2: Ensure Interface Compatibility	Goal 3: Assemble Product Components and Deliver the Product
Act-Utilitarianism	Inconclusive, depends on situation	Inconclusive, depends on situation	Support, as it benefits a number of users
Rule-Utilitarianism	Support, but depends on the set of rules chosen	Support, but depends on the set of rules chosen	Support, but depends on the set of rules chosen
Liberty rights ethics	Support, as long as the defined areas of work are respected	Oppose, conflicts with the right to work without interference within the defined limits	Support, as long as the defined areas of work are respected
Welfare rights ethics	Inconclusive, depends on the possibilities for support from different parts of the organization	Inconclusive, depends on the possibilities for support from different parts of the organization	Inconclusive, depends on the possibilities for support from different parts of the organization
Duty ethics	Support, As long as the organization has a policy that supports the goal	Support, As long as the organization has a policy that supports the goal	Support, As long as the organization has a policy that supports the goal
Virtue ethics (MacIntyre)	Inconclusive, depends on situation	Inconclusive, depends on situation	Support, as it benefits a number of users
Virtue ethics (Florman)	Support, as professionalism is stressed	Support, as professionalism is stressed	Support, as professionalism is stressed
Ethical egoism	Inconclusive, depends on amount of work required and expected additional future work.	Oppose, for the development of a specific function, this is only additional work	Oppose, for the development of a specific function, this is only additional work as the function has already be tested
Community-oriented self-realization ethics	Inconclusive, depends on situation	Inconclusive, depends on situation	Support, as it benefits a number of users

4.5 Duty Ethics

What duty ethics imply depends on the rules and guidelines developed and used in the organization. For development organization, it is often expressed as polices, indicating the expected behavior from the developers. This leads to a common conclusion for all three goals for product integration, i.e. it depends on the policy for product integration in the specific organization. However, the general idea of having a policy would be supporting the goals as long as they are a part of it.

4.6 Virtue ethics (MacIntyre)

Based on Aristotle, the virtue ethics described by MacIntyre express professions as valuable social activities. The target for the engineer would be to produce goods that can be internal or external, to adhere to standards of excellence, and to contribute to progress of the society. Internal goods can be personal (meaningful work), or public (medicine or electric power). External goods are earned through activities and include money, power and prestige.

Prepare for Product Integration. The preparation may be support as the standards of excellence is aimed for, but there may be a conflict with the aim of producing external goods as power and prestige.

Ensure Interface Compatibility. Again, this goal is basically supported, but there may be a conflict in the notion of meaningful work. Checking interface compatibility may be perceived by engineers to be unnecessary work, as they adhere to standards of excellence.

Assemble the Product Components and Deliver the Product. As for act-utilitarianism, the progress and the delivery of internal goods to the public are generally considered as good, and support this goal.

4.7 Virtue ethics (Florman)

The virtue ethics described by Florman put the emphasis on the loyalty to the employer and on professionalism, but emphasis is on desirable features rather than on expected behavior. An engineer that does the job well is a morally good engineer. The two virtues are thus loyalty and competence. To act professionally is to work according to identified and described good practices. The CMMI is a collection of good practices that has been collected from a large number of successful product development organizations. The fulfillment of the goals described can considered to be supported. However, if the organization has selected a different model, with contradicting goals for product integration, this conclusion is invalid.

4.8 Ethical Egoism

Ethical egoism focuses on long term solutions that would maximize the benefit for the individual performing the actions. Hence, care for others is not in focus, and for

an engineer developing functions for a product may not even care about the final product results. The goal would be to make sure that the individual contribution is observed as excellent.

Prepare for Product Integration. The result of the analysis is inconclusive as it depends on the amount of extra work that is the result of reaching this goal.

Ensure Interface Compatibility. To ensure the compatibility when integrating is additional work for an engineer doing development of a specific function.

Assemble the Product Components and Deliver the Product. Also for this goal, the individual engineer developing a function only sees additional work. This activity is perceived unnecessary as the individual functions have been tested in the development work.

4.9 Community-oriented self-realization ethics

Emphasis in this direction is on the commitments that individuals make, based on their self-interest, balanced with an understanding that self-realization depends on the relationships in the society. The commitments reflect what the engineer care about and govern the actions in development projects.

Prepare for Product Integration. The decisions in determining the strategy may be supported, but may also conflict with the interest of the engineer if the requirements limit the freedom for the developer.

Ensure Interface Compatibility. If this goal is to be supported by this ethical choice, the commitment from the engineer must be to follow the goal. Otherwise, the activities leading to ensuring interface compatibility will be considered unnecessary and not in the self-interest of the engineer.

Assemble the Product Components and Deliver the Product. The commitment of the individual engineer is often directed towards development of functionality in products that will contribute to society. This supports the goal of assembling and delivering the product.

5. Comparison between IEEE Code of Conduct and Different Moral Directions

To follow a code of conduct is considered to be one of the criteria for a profession to be mature [10]. For software engineering, the IEEE Code of Ethics [6] is one of the descriptions that have been developed and is also pronounced to be a sign of maturity [11]. In Table 2, the ten guidelines included in the Code of Ethics are compared to the ethical approaches that can be considered to be the basis for them. Note that an approach that does not insist on but still does not contradict the statement is not indicated below. The interpretation in this section can and should be discussed as this most likely would increase the knowledge and awareness about the influence of the ethical directions on the software engineering discipline.

Table 2. Relation between IEEE Code of Conducts and ethical directions

According to the IEEE Code of Conducts, the members should agree:	Act-Utilitarianism	Rule-Utilitarianism	Liberty rights ethics	Welfare rights ethics	Duty ethics	Virtue ethics (MacIntyre)	Virtue ethics (Florman)	Ethical egoism	Community-oriented self-realization ethics
1. to accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment	X	X			X	X	X		X
2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist Develop an integration plan based on the strategy	X	X	X	X	X	X	X		
3. to be honest and realistic in stating claims or estimates based on available data	X	X	X	X	X	X	X		
4. to reject bribery in all its forms	X	X			X	X	X		
5. to improve the understanding of technology, its appropriate application, and potential consequences	X	X			X	X	X		
6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations	X	X			X	X	X		

Table 2 (continued). Relation between IEEE Code of Conducts and ethical directions

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others	X	X	X	X	X	X	X		
8. to treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin	X	X	X	X	X	X	X		
9. to avoid injuring others, their property, reputation, or employment by false or malicious action	X	X	X	X	X	X	X		X
10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics	X	X	X	X	X	X	X		

6. Organizational influence

Many organizations explicitly select a set of guiding principles that are intended to ensure that employees base decisions on ethical principles common for the organization. However, observations made in industrial settings indicate that the influence on behavior is limited. Probable reasons for this include inadequate communication of principles and abstract definitions, but also organizational changes such as mergers, acquisitions, and lay-offs would make it difficult to convey an ethical direction to the whole organization. The individual selection will eventually determine the taken action.

7. Conclusion and future work

The influence on the effectiveness and efficiency in the workplace in general and on product integration in particular from the ethical codes followed is substantial. In most organizations, there is a mixture of different moral orientations which makes the analysis difficult. From our compilation and the reasoning above we conclude that the impact from different ethical theories is difficult to determine theoretically. An indication that a combination of several directions probably would give the best result is found through examination of different ethical codes for engineers. The gain from making ethical choices explicit is that it facilitates rational discussions and understanding of optimal choices in team work situations where different ethical

attitudes always exist, but remain un-explicated. Examples are team members that are supposed to share their knowledge, information, results, resources etc with each other, but who might follow the line of ethical egoism.

Future work should include investigations in different organizations with and without explicit ethical policies. This would increase the understanding of the influence this has on individual behavior and on product development efficiency.

8. References

[1] Martin, M.W., R. Schinzinger, *Ethics in Engineering*, Fourth edition, McGraw-Hill, New York, NY, 2005

[2] Chrissis, M.B., M. Konrad, S. Shrum, *CMMI*, Addison-Wesley, Boston, MA, 2003

[3] ISO/IEC 15288:2002, International Standard, "Systems engineering – Systems life cycle processes", ISO/IEC 2002.

[4] ANSI/EIA-632-1999, "Processes for Engineering a System", Government Electronic and Information Technology Association, Electronic Industries Alliance, 1999.

[5] ISO/IEC 12207:1995, "Information technology – Software life cycle processes", ISO/IEC 1995.

[6] IEEE Code of Ethics, www.ieee.org/portal/pages/about/whatis/code.html, (link valid April 2005)

[7] ACM Code of Ethics and Professional Conduct, www.acm.org/constitution/code.html, (link valid April 2005)

[8] Florman, S.C., "Moral Blueprints: On regulating the ethics of engineers", Harpers 257, 1978.

[9] MacIntyre, A., *After Virtue*, 2d ed. South Bend, University of Notre Dame Press, 1984.

[10] Ford, G., N.E. Gibbs, "A Mature Profession of Software Engineering.", SEI, CMU, CMU/SEI-96-TR-004, January 1996

[11] McConnel, S., *Professional Software Development*, Addison-Wesley, Boston, MA, 2004