# Case Study: Software Product Integration Practices

Stig Larsson[1], Ivica Crnkovic[2]

[1]ABB AB, Corporate Research, Västerås, Sweden
[2]Mälardalen University, Department of Computer Engineering, Västerås, Sweden
{stig.larsson, ivica.crnkovic}@mdh.se

**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.

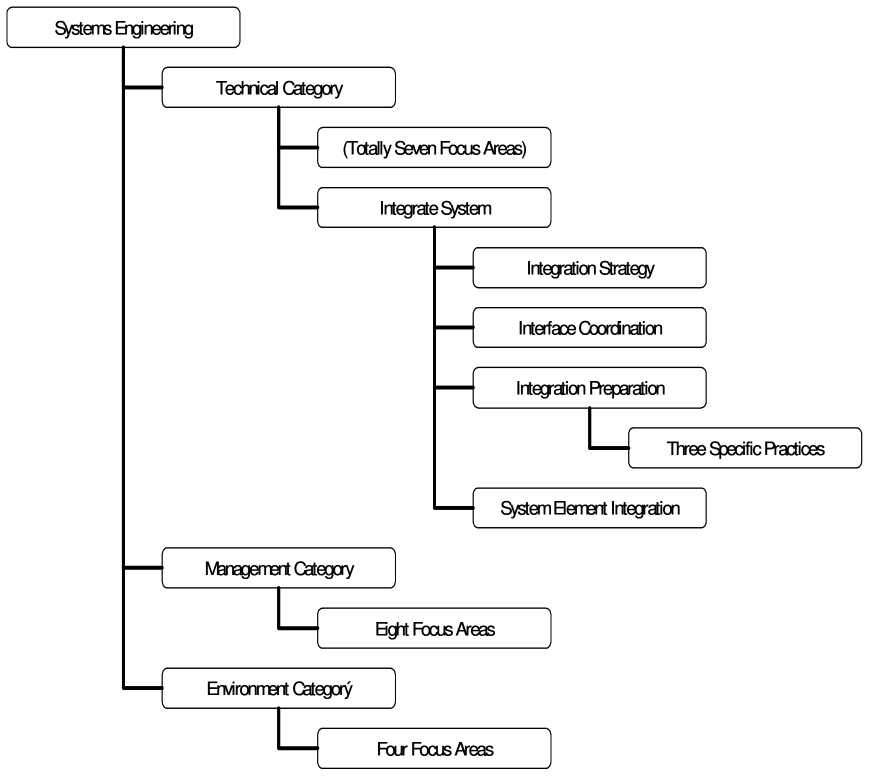

**Fig. 1.** Structure of EIA-731.1

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.

**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 | + | + | + |

## 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 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<br>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<br>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<br>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<br>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<br>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<br># of problems found | | |
|---|---|---|---|
| | Capability level 1 | Capability level 2 | Capability level 3 |
| Case 1 | 3 /7<br>5 problems | 0/4<br>No problem | 0/5<br>No problem |
| Case 2 | 5/7<br>1 problem | 2/4<br>No problem | 15<br>No problem |
| Case 3 | 7/7<br>No problem | 0/4<br>No problem | 0/5<br>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/
4. http://www.incose.org/
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)