Information and Software Technology 51 (2009) 1066–1080

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Software product integration: A case study-based synthesis of reference models

Stig Larsson [a,*], Petri Myllyperkiö [b], Fredrik Ekdahl [c], Ivica Crnkovic [d,1]

[a] ABB Corporate Research, Forskargränd, SE-721 78 Västerås, Sweden
[b] ABB Distribution Automation, Muottitie 2, FI-65320 Vaasa, Finland
[c] ABB Robotics, Hydrovägen 10, SE-721 68 Västerås, Sweden
[d] Mälardalen University, Department of Computer Science and Electronics, P.O. Box 883, SE-721 23 Västerås, Sweden

ARTICLE INFO

Article history:
Received 6 July 2007
Received in revised form 11 November 2008
Accepted 5 January 2009
Available online 23 February 2009

Keywords:
Product integration process
Process improvement
Process validation

ABSTRACT

In software intensive systems the integration becomes complex since both software and hardware components are integrated and run in the execution environment for the first time. Support for this stage is thus essential. Practices for Product Integration are described in different reference models. We have investigated these and compared them with activities performed in seven product development projects.

Our conclusion is that descriptions of best practices in product integration are available in different reference models, but need to be merged into one set of practices. Through case studies we see that the described practices are insufficiently used in industry, and that organizations would benefit from adhering to them. Our investigations indicate that a set of practices are necessary to be successful in software product integration: define and check criteria for integration, review interface descriptions and ensure coordination of interface changes, and deliver components as agreed. In addition to these, a set of practices are supporting the integration activities, including the definition of an integration strategy, and the establishment of a suitable integration environment.

1. Introduction

Integration of software products, as well as products that include software, is described in several standards and other collections of best practices, i.e. reference models. Even if the product integration process is included in many development process models and part of the development iterations, the process is in many cases a distinctly recognizable process. In particular, the development of complex systems in a distributed environment where components are developed in different locations requires a coherent, comprehensive and separate description of the integration phase.

The practices in the different reference models are compiled from experiences from different companies and organizations. Practices are selected as they are considered to increase the effectiveness and efficiency of the product development as well as contributing to the quality of the product. The source for the described practices is collective experiences but the resulting documents are rarely validated through independent research. This article is an attempt to investigate product integration practices, compare the reference models with experiences from different development projects, and to aim at a first step of validation.

In order to define the context and scope for this paper, we use the definition of integration for product and system development found in the glossary of EIA 731.1 (interim standard) [11]:

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

In spite of the existence of many reference models the problems in integration persists. Investigations of different organizations developing products [4,31,34] as well as our own experience tell us that product integration often is where problems occur. The difficulties found during integration include mismatch between the different components, problems with properties of the system (e.g. performance, response time) that are observable first after integration, and problems in other parts of the system than the one that was changed or added. To better understand this problem we have analyzed the integration processes in two companies and a total of seven product development projects from different organizations within the companies. As a starting point for our study we have stated the following questions:

- How are the practices that are described in reference models useful for product development units aiming at improvements in the product integration process?

* Corresponding author. Tel.: +46 21 345135; fax: +46 21 32321.
E-mail addresses: stig.bm.larsson@se.abb.com (S. Larsson), petri.myllyperkio@fi.abb.com (P. Myllyperkiö), fredrik.x.ekdahl@se.abb.com (F. Ekdahl), ivica.crnkovic@mdh.se (I. Crnkovic).
[1] Tel.: +46 21 103183.

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.01.001

- What is the core set of practices that can be identified to reduce problems in product integration?
- Is it appropriate to combine reference models to provide better support to product development units, and how can this be done?

Our approach to these different perspectives is to study the performance of the process in the investigated organizations and compare the activities with the ones prescribed in the reference models regardless of the development model used. We also look at the problems in the organizations and analyze these with respect to the practices proposed in the reference models.

We claim that we by investigating a number of projects and the practices in use have been able to determine to which extent the practices described in reference models are useful as a support for development units. We also claim that we through the investigation can identify a need for revisions of the reference models.

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

However, we find that no reference model provides full support for the product integration. By combining them a significant improvement can be made. For this reason we propose a more complete, consistent and integrated combination of the described practices.

The terms "product integration", "systems integration" and "software system integration" are used for several different aspects in product and system development literature. Djavanshir and Khorramshahgol [9] have investigated the importance of different process areas related to system integration and observe that professionals in the field relate integration to many areas of systems engineering. This indicates that there is no clear definition of integration when discussing system and software engineering. It is consequently necessary to clearly define the scope of integration, and to be aware of other interpretations of the term. Sage and Lynch provides an overview in [36]. The main uses of the terms are:

- Product integration processes
  This term describes the process used in product development projects when parts are combined into more complex parts and eventually into the product or system to be delivered to the customer. It includes the activities ensuring that the combinations of components are functioning as intended, and the management of interfaces between components as well as between the product and the environment. As earlier described, this is the focus for this article.
- Architectural, or technical, product or system integration
  This concerns the technical solutions used to fulfill requirements on functionality and quality attributes such as reliability and performance. Different levels of integration include export and import facilities, the use of wrappers and adapters, integration through shared databases, and integration on source code level. Interface design is one important issue for all levels of architectural integration, and standard interfaces are available for many applications. Examples of the use of integration in this meaning is found in [15] where Garlan describes trends in software architecture research, and in [16] where Gorton describes useful architectural practices.
- Enterprise Application Integration (EAI)
  EAI is a type of architectural integration where organizations combine and integrate existing and new systems to assist the organization in achieving business objectives. This type of integration is performed to ensure data consistency and to make information accessible to different types of stakeholders, often based on the use of a common middleware. Examples of descriptions of EAI are [7] by Cummins, [32] by Linthicum, and [35] by Ruh et al.
- System and product interoperation
  Interoperation is a looser type of connection between products and systems. The integration process for systems that need to interoperate is similar to the product integration described here, but has not been further investigated in this study.

The remainder of this paper is outlined as follows: Section 2 refer to related work. Section 3 describes the research method used in the studies. Section 4 provides an analysis of product integration included in different reference models. Section 5 describes the case studies and presents the data from these. Finally, in Section 6 we discuss the results of the studies, give conclusions, and propose further research based on the results.

2. Related work

Product integration processes are included in different established reference models for product and system development such as ANSI/EIA -632 [2], EIA-731.1 [11], ISO/IEC 12207 [23], ISO/IEC 15288 [24], and CMMI [39]. They can be used as a source for assessments and process improvement planning. Reference models are normally articulated as a set of requirements on the development process, and not as a set of activities to be performed, giving the organization possibility to implement a suitable process. One concern with reference models is the inadequate validation of them; it is difficult to find research examining the validity of the content of these reference models. However, the reference models are developed based on experience from a large number of organizations, making it likely that important considerations are taken into account.

A general description of system and product integration has been made by Sage and Lynch [36]. The paper describes various views including life cycle, architecture, process, interfaces, and enterprise integration. The roles of architectures and integration in different reference models are also described. One conclusion is that methodologies and tools for system integration and integration architectures were not well described in literature at that point in time.

In [6], Chittister and Haimes argue that the system integration process is additionally complicated when software is included, and propose that the risk factors along the software development life cycle must be identified and understood through measurements and ultimately mitigated.

Stavridou has examined product integration from two different perspectives. In [42], integration standards for critical software intensive systems are investigated. The examination focuses on military policies and standards, but includes ISO/IEC 12207 in the comparison. The conclusion of the analysis is that the majority of the examined standards address integration testing, but that the standardization is not appropriate for many integration issues, and that additional guidance for the project manager is needed. A more technical approach is selected in [41] where the integration is proposed to be considered as a design activity.

The activities in the product integration area have also been the subject of interest from the agile community where frequent builds is one of the cornerstones. One example is Fowler [14] who describes the requirements on developers: before committing back to mainline the developer would need to update his work area with the latest mainline, i.e. build against the latest changes of other developers. Only after that, integration into the mainline would be permitted.

Eppinger describes in [41] a method to reduce the problems in integration using an architectural and design structure matrix approach. The method includes three steps: decomposition, identification of interactions between the components based on different types of interaction, and clustering of components based on the analysis of the structure of interactions. The method is closely related to the management of interfaces as described in product integration.

Schulte describes the integration of large systems as a challenge and proposes a method for handling uncertainties in the resulting system characteristics when integrating components [37]. Another example of using models to ensure efficient and effective integration has been presented by Karsai et al. [25]. The point made is that modeling should be made the central activity when developing systems. De Jonge [8] identifies that the integration of components is more difficult when reusable building blocks are applied and proposes techniques that promote fine-grained software reuse.

In addition to using reference models to improve product integration, metrics can be used. Houston has studied the integration problems occurring in an avionics system [17]. Different types of integration problems were identified based on the functionality in which the problem occurred. By classifying the integration problem reports and estimating the expected handling time through simulations of the report types several possible improvement activities were identified. The study complements our research as one alternative route to find improvement activities. Chiang and Mookerjee describe a method using metrics to identify when it is suitable to start the integration activities based on fault detection in the module development, introducing a threshold concept [5].

Four different aspects of systems integration is described by Nilsson et al. in [33]. The paper addresses the technical characteristics of integration including integration technology, integration architecture, semantic integration, and user integration. The main message from this standpoint is that systems integration is difficult and complicated and that there are no obvious shortcuts.

Kuhn concentrates in [26] on effective use of standards for interfaces when integrating systems, and describes a methodology to application development that focuses on an architectural approach.

Component-based software engineering is also considered to simplify the integration if taken to the extreme. In [10], Dogru describes a fully component-oriented approach and puts this in contrast with modifying object-oriented approaches, stressing that component-based development leaves out inheritance and capitalizes on composition. However, the lack of tools and experience currently prevent full use of the presented ideas.

Incorrect use of reference models and software development models is described by Fitzgerald [13]. The reason that organizations are not using the models as intended is claimed to be the perceived lack of contribution to successful product development from the models. Another reason is the perceived inflexibility in the models, not allowing for customization to specific organizational and project needs. This has also been highlighted by Bajec et al. in [3]. They describe and prescribe a method for adapting the development model to each specific project. This should of course also include the product integration part of the process.

The research in this article build on the work published as separate case studies. Case 1 is described in [29] and as case 1 in [30], Case 2, 3 and 4 are described as case 1, 2 and 3 in [28], and Case 6, 7 and 8 as case 2, 3 and 4 in [30]. An early version of the summary of the reference models described in Section 3 can be found in [27]. The research in this paper summarizes the data from the case studies, and a new broader analysis has been made, including more reference models. One result is a compilation of what practices related to product integration are included in the different reference models. The combination of that analysis and the data from the case studies has enabled us to have stronger indications on what practices are needed for efficient and effective software product integration.

3. Research method

Our study includes experiences from seven product development projects run in five organizations in two companies. The products developed in these organizations include applications from manufacturing industries, process industries, telecommunication, power distribution, and power transmission. Both companies are multinational with development in many countries. The experiences from the investigated product development organizations are compared to and classified according to a set of standards and models, i.e. reference models. This is done through three activities: investigation of reference models, data collection, and a mapping between the reference models and the data.

Two types of reference material, standards and models, have been considered in this study and are referred to as reference models. The difference between the types is that standards have been approved by a standardization body, while a model may be issued by any company or organization. The included reference 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 reference models is based on available information from standardization organizations such as ISO [22], ANSI [1] and IEEE [19] and references from organizations such as SEI [40] and INCOSE [20]. Based on the focus of our research, product integration in development of products that include software, two specific selection criteria have been used in the choice of reference models:

(i) The reference model should be relevant to development of products that include software.
(ii) The reference model should include requirements on product integration, implicitly or explicitly, as this is our research area.

The standards provided by the listed organizations have been evaluated based on both. The reference models that have been selected are:

- ISO/IEC 12207 Information technology – Software life cycle processes [23].
- EIA-632 Processes for Engineering a System [2].
- CMMI Capability Maturity Model Integration [39].
- EIA-731.1 Systems Engineering Capability Model [11].
- ISO/IEC 15288 Systems Engineering – System life cycle processes [24].

Also ISO 9001 [21] and IEEE Std 1220-2005 [18] were considered, but for both these standards the expectations on the product integration process are limited, and they have consequently not been further analyzed. It should also be noted that efforts are made within the standardization bodies to harmonize several of these reference models such as IEEE Std 1220-2005 [18], EIA-632 [2], ISO/IEC 15288 [24] and ISO/IEC 12207 [23].

To accommodate the comparisons between the different reference models practices related to product integration has been extracted and listed in Table 2. This is done through a detailed review of the complete reference models based on the following definition of product integration found in CMMI [39]:

"The scope of this process area is to achieve complete product integration through progressive assembly of product components,

in one stage or in incremental stages, according to a defined integration sequence and procedures. Throughout the process area, where we use the terms product and product component, their intended meanings also encompass services and their components.

A critical aspect of product integration is the management of internal and external interfaces of the products and product components to ensure compatibility among the interfaces. Attention should be paid to interface management throughout the project."

The definition in CMMI has been selected as it explicitly defines the scope of product integration. The findings have been documented as comments in the standards and compiled into Table 2. In the table, references to the chapters in the reference models show where to find the original information. Each practice has been described through a general statement that also is used as a headline for the description of that practice.

The case studies were designed based on the methodology described by Yin in [43]. This includes the preparation and the implementation of the studies through interviews and document reviews, and the analysis based on the observations.

For the data collection, four main questions have been formulated. Based on these, the questions for the interviews have been expressed. The four main questions are

- How is the preparation for product integration performed?
- How are interfaces managed?
- How is the actual integration of the product performed?
- What types of problems have been observed in relation to the product integration?

The data collection is based on interviews and document reviews through appraisals of projects and organizations.

The interviews were based on a set of detailed questions derived from one or more of the reference models. The researchers were guided by a discussion guide for each case. The specific questions in the discussion guide varied between the studies depending on the reference model that was used as a basis for each study. The interviews have been open-ended discussions about the integration process. The interviews were made either in groups with interviewees representing a function or individually if a function was performed by a single person. During the discussions, the researchers have ensured that all questions in the discussion guide were covered. Each interview lasted between one and two hours. Document reviews were performed on the documentation describing the integration process, the training material for the organization as well as files used for and as a result from the product integration process. Examples of document types that have been examined are project plans, configuration management plans, integration plans, quality assurance plans, build reports, integration reports, and test reports. Besides an understanding of how the process is performed, information about the documented as well as perceived problems related to the product integration process was captured through explicit questions about issues in the integration. The documentation from the data collection consists of notes taken during the interviews and samples from the written documentation.

More details on the interview method is described by Ekdahl and Larsson in [12]. The requirements for different classes of CMMI-based appraisals are described by SEI in the ARC [38]. The appraisals were performed as Class C appraisals also when the basis for the data collection was other reference models than CMMI. For four of the cases (1, 5, 6, and 7), data from the execution of the production integration process has also been collected, i.e. in the form of build data.

Additional information was collected from the product integration process in case 1, 5, 6, and 7. This information includes problems and failures in the build, smoke test and regression test activities, and is further described in [30]. The primary data from builds and tests has been collected by the practitioners and compiled by the researchers. Additional information regarding the problems has been added by the practitioners, i.e. notations from the integration teams regarding errors and faults discovered in the build process have been documented.

The final activity in our research has been to map the findings from the interviews to different reference models. A first mapping from the cases to the reference models was made to find out what practices were performed in each organization. This was done for each practice through searching the collected material for evidence that the practice was performed. This way all the practices were covered and additional information about the organization besides the practices were captured.

A second mapping was made to understand how the problems found in each case relate to the practices. This was made for each problem through searching the collection of practices for a match. Problems that could not be related to any product integration practices were noted and discussed, but have not been used further in this study.

Care has been taken to ensure that all classification is determined by two different researchers. In cases where the researchers have drawn different conclusions, a discussion has been held to clarify the different opinions, and an agreement has been sought. If an agreement or conclusion has been impossible to reach, this has been clearly indicated in the presentation of results as being undetermined.

4. Practices in standards and models

Each of the selected reference models is in this section described starting with the purpose and intention of the model and continued with details on the description regarding product and software integration processes. The actions and tasks considered to be related to product integration are summarized. Note that these summaries are for information purposes only, and that the original text in the reference models should be used for any implementation.

Based on the acquired knowledge regarding the reference models, a summary of practices and a comparison between the models have been made. The purpose has been to see if there is a set of practices consisting of the union of the used reference models.

The first step was to combine the extracted information from all investigated reference models into a set of practices. After that, all reference models were investigated based on the set of practices. Both explicit and implicit instances of the practices were noted. This classification is relying on the experience and knowledge of the researchers. However, through the stepwise approach, the risk for missing information is reduced as each reference model is examined twice.

4.1. 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 [23]. 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. Note that the update of user documentation is omitted in this investigation as it is not considered to be a part of product integration.

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.

4.2. EIA-632

The purpose of the EIA-632 standard [2] 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.

4.3. 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 [39]. 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.

4.4. EAI-731.1

The purpose of EIA-731.1 (interim standard) is to support the development and improvement of systems engineering capability [11]. It 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.

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

ISO/IEC 15288:2002 is intended to describe the life cycle of systems [24]. 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 include 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.

4.6. Summary of reference model practices

Table 2 summarizes the product integration process as described in different reference models and provides a basis for comparison. In this section, we have extracted practices described in each reference model and combined each of these practices. The combination of the selected reference models has given us 15 different practices that have been expressed in a general way. These have been selected as they are *directly* related to product integration. Other practices mentioned in the context of product integration in the different reference models have been excluded. Examples of this are CMMI, Specific Practice 3.4, "Package the assembled product or product components and deliver it to the appropriate customer", and ISO/IEC 12207, Section 5.3.8.3 "The developer shall update the user documentation as necessary". The excluded practices are important, but have been considered to be only tangentially related to product integration as defined for this paper.

The proposed practices for each of the activities are described below, and further guidance can be found through pointers to the reference models in Table 2. Note that the practices are not ordered in the sequence that they are expected to be performed. Most of the activities described should be carried out in an iterative manner. However, our advice is to ensure that Product Integration (PI) Practice 1 is performed early in any product development project to set the expectations on the remaining practices.

4.6.1. PI Practice 1. Define and document an integration strategy

Develop an integration strategy and supporting documentation that help in identifying a sequence for the product integration satisfying the product requirements while minimizing risks. The documentation should include different considered strategies, and the rational for selecting one. The strategy should be based on factors important for the product development such as the need for partial systems, verification and validation strategies, organizational arrangements and selected architectural solutions. As the development proceeds, the strategy should be reviewed periodically to ensure that the foundation for the decision is still valid.

Examples of strategies are to start with platform functions to simplify the addition of many applications in parallel, to ensure customer functionality to be added early enabling demonstrations, or to have continuous integration of product components as they become available. Each of these strategies has advantages and drawbacks that must be understood. The strategy will affect the

planning for the whole project, and is essential for the understanding, planning and preparation for product integration.

4.6.2. PI Practice 2. Develop a product integration plan based on the strategy

The product integration plan should define the integration steps as an assembly sequence, the procedures to be used for integration, the integration verification to be performed, resources, and responsibilities. Based on the selected strategy, the plan may include alternate sequences to minimize risks and prepare for different scenarios. The plan should be reviewed periodically and updated as needed based on new information and risks.

4.6.3. PI Practice 3. Define and establish an environment for integration

The product integration plan will together with the product requirements provide requirements on an integration environment. The definition and establishment of the environment can be reused from organizational resources, developed, or acquired. In either case, the requirements and plans for the environment need to be considered in parallel with the development and integration plans.

The integration environment typically consists of simulators, stubs, test equipment, parts of existing components and products, software and hardware tools, and measurement equipment.

4.6.4. PI Practice 4. Define criteria for delivery of components

The criteria defined for delivery of components should be selected so that they can indicate the readiness of a component for integration. The criteria should address what type and level of verification should be performed on the component and interfaces as well as thresholds of the verification results for acceptance.

4.6.5. PI Practice 5. Identify constraints from the integration strategy on design

Several factors can be considered when identifying the constraints that a specific integration strategy may impose on the design. These include rules for what types of interfaces should be available to enable interconnection between components at different stages of the integration. Also the necessity to use simulation, stubs, and wrappers need to be considered as this may require specific solutions. The environment used for the integration may require that the design is constrained.

Typically, the constraints on architecture and design due to a specific integration strategy require a revision of the architectural and design documentation.

4.6.6. PI Practice 6. Define interfaces

The efficient and effective execution is depending on interfaces that are agreed and used for the different components. This includes physical, functional, and logical interfaces. When interfaces are determined and defined, an agreed set of criteria should be followed. The criteria typically expose attributes important for a specific application, and may include parameters reflecting the requirements on dependability, performance, safety, evolvability, and other quality attributes. Once determined, the interfaces should be documented and put under configuration management. This documentation should include the rational for the selected definition and design. The interfaces are typically characterized through the source, destination, control and data characteristics for software, and electrical and mechanical characteristics for hardware. Also human interfaces and environmental parameters should be addressed and documented.

Early definition of interfaces reduces the risk for mismatch between product components that are developed in parallel. The drawback is that knowledge is acquired as the implementation of the product components progresses; additional interfaces may be needed, as well as modifications to existing ones.

4.6.7. PI Practice 7. Review interface descriptions for completeness

When interfaces are defined and revised, appropriate stakeholders need to perform a review to ensure that each interface description is complete and fulfill the intentions as described in the requirements. This is however not sufficient. As the product components are developed, there is a need to review the interface descriptions periodically to ensure that they are sufficient and understood by all stakeholders. These reviews should also provide input to proposed interface changes.

To facilitate proper reviews, interfaces categories can be defined. The definition of the categories can be used to decide on what needs to be documented for each category. Once established, the categories can be used to organize the interfaces, and made available to relevant stakeholders.

4.6.8. PI Practice 8. Ensure coordination of interface changes

Interfaces affect different stakeholders, and the changes must be controlled to reduce misunderstandings as well as late discovery of mismatch. Changes should be controlled for different types of interfaces, e.g. between product components, to the environment, to users, and to verification equipment.

Change Control Boards can be set up to control changes to interfaces. This is critical for projects that have external suppliers, or depend on other parts of the organization. The responsibility of the Change Control Boards goes beyond deciding on changes; consequences should be investigated before decisions are made, relevant stakeholders should be involved, information regarding decision on changes should be communicated, and the interface documentation updated as appropriate.

4.6.9. PI Practice 9. Review adherence to defined interfaces

As a product component is to be delivered to integration, compliance to the interface documentation should be reviewed and verified. The criteria used for definition of the interfaces can be used as support for the review.

A review of interface adherence may be done in a common session for product components using a specific interface. This enables the development teams to agree on any mismatch and decide on changes to one or several of the components, or propose a change of the interface.

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

The requirements considered for the integration tests are the ones related to interfaces and interaction with other components and the consistency with the architectural design. It is important the specified tests are in line with the verification strategy, i.e. focus on the areas defined for integration testing.

4.6.11. PI Practice 11. Verify completeness of components obtained for integration through checking criteria for delivery

Each product component to be integrated must be identified as being the intended one in the right version. The completeness of a component can only be confirmed through checking defined criteria. If a component does not fulfill the criteria appropriate measures should be taken; changes may have to be made to the component, or the deficiency can be accepted temporarily or permanently. If accepted permanently, appropriate changes should be made to requirements and other documentation.

The responsibility to ensure that a product component meets the defined criteria can be decided in the strategy for product integration. Typically, the developer or development team is responsible to develop the product component in accordance with all

requirements, including the criteria for integration. However, it is also common that the integration team is responsible to check are that the criteria are met, and to reject the delivery of components not adhering to the requirements. Handshake procedures are suggested to reduce tension between different teams, ensuring a common view of the situation.

4.6.12. PI Practice 12. Deliver/obtain components as agreed in the schedule

As the product components are verified as complete as defined by the product integration delivery criteria, they can be delivered for integration. It is of utmost importance that any slippage in the agreed schedule for the delivery of a component is communicated as soon as it is know, or even as soon as the risk for late delivery is identified. Any delays may affect the integration sequence, and the ability to provide different stakeholders with intermediate integrations, and with the final product. If the delays are know early, countermeasures can be initiated.

The acknowledgement of reception for integration is important, and can be made through an informal or formal handshake procedure. An example of this is to use the configuration management status information to set the product components in different states. This enables relevant stakeholders to get an understanding of the status, and bottlenecks can be identified.

4.6.13. PI Practice 13. Integrate/assemble components as planned

The integration and assembly of the components should be performed as described in the product integration plan. The integration can be made in steps, with aggregates of components being built consecutively, and it may be necessary to perform evaluation activities on the intermediated results. The result of the assembly should be made available to all relevant stakeholders.

4.6.14. PI Practice 14. Evaluate/test the assembled components

The focus when evaluating the assembled components is on interface verification. The defined and described procedures and environments are used to ensure that the product components work as intended when combined. The results from the verification should be recorded and appropriate action taken to handle any issues that may occur.

4.6.15. PI Practice 15. Record the integration information in an appropriate repository

When the integration is performed, it is necessary to record information regarding problems in the product, product components, integration environment, and in procedures for integration. The information can, besides a control of necessary changes to the product and product components, be used to further improve strategies, practices, environment, and process improvements for product development processes that are delivering to the product integration.

Examples of information that can be collected are problems in the integration related to different practices, e.g. build statistics [30].

The list of activities can be used as a guideline for the definition of a product integration process and process improvement in the area. Note that if a reference model is implemented, the original text for that specific reference model should be used.

In the compilation of the practices from different reference models in Table 2, three different types of indications have been used. "***E***" is used if the practice is explicitly described in the reference model, "***I***" if it is implicitly described and a "**–**" if it is not described. Implicit descriptions are identified if there is a generic statement that the type of activity, such as reviews, should be performed. If a practice is covered both explicitly and implicitly, only the explicit occurrence is mentioned in the table. A pointer to the reference model is given for each explicit or implicit description of the practice. The references in the table are numbers of sections, practices, or requirements as defined in Table 1.

Table 1
References for the different reference models.

Reference model	Reference
ISO/IEC 12207	Section
EIA-632	Requirement
CMMI	Specific Practice in the Product Integration process area
EIA-731.1	Specific Practice
ISO/IEC 15288	Section

4.7. Similarities and difference between the reference models

A comparison of the standards based on the PI practices show that there is an on-going development of the area and an increased agreement over time on what can be considered to be best practices. The following observations have been made:

- Integration planning is expected in all reference models
- Only ISO/IEC 15288:2002 mention the aspect that the integration strategy may imply constraints on the system or product design
- Interface definition is explicitly specified in all reference models except ISO/IEC 15288:2002, but other aspects of interface management such as review and control of changes are only specified in CMMI and EIA 731.1
- The verification of completeness as well as the actual integration and verification of the assembled components are included in all reference models.

The comparison between the different reference models indicates that expectations on the preparation for integration and the handling of interfaces have been made more explicit over time; additional practices are added and already existing practices are made more precise for reference models released at later dates.

Older standards are less explicit regarding product integration, while newer focus on different aspects. As EIA has been used as an input to the development of CMMI, there is no surprise that they are handling product integration in a similar way. ISO/IEC 15288 has the best coverage of product integration except for management of interfaces.

Our conclusion is that additional investigations and comparisons are needed to understand how the area evolves, what factors are determining what is added to the reference 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.

5. Case studies

In order to understand if the reference models can help organizations reduce the problems in product integration as executed in an industrial environment, we have examined seven different projects. All of the projects were initiated to develop products used in the manufacturing, process, telecommunication, or power domains. This section describes the projects and products for each case. One notable characteristic is that the projects in both companies are to a certain extent independent in their selection of work processes and supporting tools. This is a strategic decision based on the diverse needs from different types of development and products in both the investigated companies.

Table 2
Product integration process in selected reference models.

Reference models	ISO/IEC 12207	EIA-632	CMMI	EIA-731.1	ISO/IEC 15288
Publication date Generic activity description	Aug 1995	Jan 1999	Mar 2002	Aug 2002	Nov 2002
1. Define and document an integration strategy	-	I Req 32 a	I PI SP 1.1	E SP 1.5-1-1SP 1.5-1-2	E 5.5.6.3a
2. Develop an integration plan based on the strategy	E 5.3.65.3.75.3.8	E Req 32 a	E PI SP 1.1	E SP 1.5-1-3a	E 5.5.6.3a
3. Define and establish an environment for integration	-	I Req 32 a	E PI SP 1.2	–	E 5.5.6.3c
4. Define criteria for delivery of components	I 5.3.8	I Req 32 a	E PI SP 1.3	I 1.5-3	I 5.5.6.3e
5. Identify constraints from the integration strategy on design	-	-	-	-	E 5.5.6.3b
6. Define interfaces	E 5.3.45.3.55.3.6	E Req 16 bReq 17 b	E TS SP 2.3	E SP 1.3-1-1c SP 1.3-1-3a SP 1.5-2-3a, 3b, 3c	I 5.5.4.3g
7. Review interface descriptions for completeness	I 5.3.5	I Req 12 d	E PI SP 2.1	E SP 1.5-2-2a, 2b	I 5.5.4.3g
8. Ensure coordination of interface changes	-	I Req 12 d	E PI SP 2.2	E SP 1.5-2-1a	I 5.5.4.3i
9. Review adherence to defined interfaces	-	I Req 12 d	E PI SP 2.2	E SP 1.5-3-1a	E 5.5.6.3f
10. Develop and document a set of tests for each requirement of the assembled components	E 5.3.65.3.7	E Req 32 a	I PI SP 1.3	I 1.6-2	I 5.5.7.3e
11. Verify completeness of components obtained for integration through checking criteria for delivery	E 5.3.8	E Req 3 b Req 20 b	E PI SP 3.1	E SP 1.5-3-1a	E 5.5.6.3e
12. Deliver/obtain components as agreed in the schedule	E 5.3.8	I Req 20 a	I PI SP 3.1	E SP 1.5-3-2	E 5.5.6.3d
13. Integrate/assemble components as planned	E 5.3.85.23.106.4.2	E Req 20 c	E PI SP 3.2	E SP 1.5-4-1a	E 5.5.6.3f
14. Evaluate/test the assembled components	E 5.3.96.4.2	E Req 20 dReq 32 b	E PI SP 3.3	E SP 1.5-4-1b	E 5.5.7.3e
15. Record the integration information in an appropriate repository	I 5.3.95.3.10	-	I PI SP 3.3	I 1.5-4	E 5.5.6.3g

In each of the cases we have captured the problems appearing in product integration. A problem is a reoccurring reason for failure in the integration process. This includes problems in the build, smoke test, and regression testing. For all cases problems have been captured in the interviews and document reviews. In case 1 and cases 5 through 7, also measurements from the build and integration test phases have been used as a source for finding problems and their causes. All problems are summarized in Table 3.

The two companies are multinational, with development organizations distributed globally and with most of the product development performed in Europe. Company A has more than 100,000 employees while company B has more than 60,000. Case 1, 4, 5, 6, and 7 are from company A while case 2 and 3 are from company B.

5.1. Case 1

This study was performed at a unit developing industrial control systems belonging to company A. 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.

The implementation consists of approximately 2500 KLOC of C language source code divided in 400–500 components, organized in 8 technical domains. The development organization for this product is around 100 persons. The system has a layered architecture and component-based design within the layers. The software platform defines an infrastructure that provides basic services like a broker for message-based inter-task communication, configuration support, persistent storage handling, system startup, and shutdown.

The development is performed in subprojects with responsibilities for the different technical domains based on common requirements. This results in a need for coordination of the implementation of functions needing solutions in more than one technical domain. Also project coordination is crucial to ensure that the right functions are planned and implemented for each integration point.

Table 3
Problems captured in the case investigations.

Case and problem	Problem description
1 A	Functions are not always delivered in time for integration or may be incompletely delivered. In addition to this, delivery is complicated through two different ways to deliver code. This leads to problems in the build process or in integration and system tests
1 B	Functions are not tested as required by the developers. This leads to problems in the builds and in initial integration testing
1 C	Changes in common resources (e.g. common include files) are not controlled. This results in errors appearing in other components which have not been changed
2 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
2 B	Errors are corrected that should not be. This results in new errors with higher influence on functionality and performance
2 C	Errors appear in other components which have not been changed
3 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 A	Scattered architecture on the server side as a result of the decision to handle communication in each component
5 A	Inconsistent code is delivered, and files are not included in the build as planned. The result is failed builds
5 B	The build environment sometimes contain traces of earlier builds or fail due to unstable applications used, resulting in failing builds
6 A	Changes are sometimes untested before integration, resulting in errors in initial integration testing
6 B	Files are not delivered to integration as planned and required, resulting in errors in the build process
6 C	The build environment contains sometimes traces of earlier builds, resulting in failing builds
6 D	Changes are made to interfaces without proper control. This leads to errors in the builds or initial integration testing
7 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
7 B	Untested changes to build scripts are introduced
7 C	Errors appear in other components which have not been changed

The integration is made throughout the project at integration points, with a development iteration performed for each integration point. The time between two integration points is between 10 and 12 weeks. Functionality to be included in integration is defined at the start of the iterations. Revisions of content may occur based on the progress of the project. The procedure for integration starts with the increased control of what is allowed to be included in the daily builds. When the integration point is reached, a frozen version of the system is made available for verification and validation in different parts of the organization. All errors found are considered for correction which is then included in the next integration.

The product integration process was investigated through a CMMI Class C appraisal and through investigation of build results. Two of the problems described for this case, 1A and 1B, are reported by the build and integration team and are found at build time or when performing the smoke tests. Problem 1C is reported by the verification team and the developers working with further development based on a frozen version. All problems found in case 1 are related to the coordination: functions not being delivered as promised, functions not being tested before delivery, and that changes of common resources are not controlled. This organization is now putting more effort in place into ensuring that functions are tested before delivery and to control interface changes.

5.2. Case 2

The organization in case 2 is part of company B. The product 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 defined and documented 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 to allow incremental development. 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 data for case 2 were captured through a Class C appraisal, and was based on a discussion guide developed to be neutral in relation to the reference models. Three problems have been captured for case 2 and have been reported both by the verification team and the developers. 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 explicitly described and documented generate this problem.

5.3. Case 3

The organization in case 3 belongs to company B. The group develops a product that includes software close to the hardware. The target system includes a complex hardware solution with the application divided on two target systems.

The development group is around 10 persons and follows a documented development process. The requirement specification is analyzed by the architect. The architect decides on the implementation solution, which is handed over to the developers. The developers deliver to integration at the end of the implementation phase. The 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 tests are available. The architect is responsible for implementation decisions. Typical time for the development of a release is 1.5 years. This includes the full development cycle from defining the requirements to system testing. The integration is for this system a one-time effort for each release. The rules for the integration defines what should be ready before the functional tests, not what needs to be ready before check-in.

Data was collected through a Class C appraisal based on a model-independent discussion guide. The problems reported for this case appear because of the incapability and version mismatch of the test system, the final product and the test and final hardware platform. All problems are described both by the architect and the verification responsible. 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.

5.4. Case 4

The development organization in case 4 belongs to company A, and 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, and most developers participated in the investigated project.

In recent years the current architecture has been 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 implement different parts of the user interface. Each component handles the communication with the server. This design was used to allow 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 that is used permits the developers to easily isolate problems and to minimize the uncontrolled interference and dependencies between the components.

The development is done based on 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. The continuous integration includes build and testing. The testing is partially automated. Releases are made monthly, and are verified by a separate team. There are no specific routines in place for handling the interfaces. Changes are in practice always checked by the system architect.

The Class C appraisal performed in this case was based on the same discussion guide as used for cases 2 and 3.

The new system design has reduced the implementation time for a function by two thirds. 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. The problem reported for this case is related to the technical solution; the server side implementation is scattered which has lead to inability to check the interfaces for completeness. This problem was reported by both the architect and the responsible line manager.

5.5. Case 5

The organization in case 5 belongs to company A. It develops a complex real-time control product including event, trend and error handling, data collection, communication, and operator interface. The product is part of a suite of about 30 products, forming a system that is used in process industries. The development is tightly coordinated with the development of these other products. The organization is the largest in the investigation and the number of developers involved is close to 80. The development process varies between different groups in the organization, but all parts are delivered to the build and integration process.

The product consists of more than 3000 KLOC and consists of applications on a workstation. The architecture is distributed and care has been taken to define different layers to achieve separation-of-concerns.

The project that has been investigated is the integration project for a major release program. The integration project consists of a build team and a team responsible for the automated regression tests. These two teams have possibilities to capture integration problems and to analyze background problems. The builds are performed as daily builds, weekly builds and baseline builds. The daily builds are performed to ensure the stability of the code base, and have the build environment adapted to additions in functionality. Weekly and baseline builds are made available to other parts of the organization for verification purposes. Development cycles are between 12 and 18 months.

The organization appraisal was done as a CMMI Class C appraisal. In addition to this, the build statistics were analyzed. The problems have been captured and reported by the build team.

Due to the size of the system, the build and integration environment is fairly complex. As a result, the major problems in the integration project are failures due to the environment. However, also the delivery of inconsistent code causes problems. The major reason for this is the lack of control of what is delivered for integration.

5.6. Case 6

The project in case 6 develops embedded software for the new generation of protection relays for electrical networks and belongs to company A. The development organization is small with around 15 developers. The main development team is located in one site, but a few persons from other sites have participated in the development.

At the time of the study the size of the software was around 500 KLOC. However, the final products built on the technology will be larger than that.

The product architecture consists of subsystems that are base software, application components and higher level services, such as communication services. The base software is a platform that provides the running environment for the application components and services, and separates them from the hardware. Application components are modules that each implements certain protection functionality in the final product. The functionality of the product can thus be defined by selecting appropriate application components that together with the selected communication protocols and other services fulfill the functionality required by the customer. Application components and services are having interfaces to the base software, but not to each other. While these subsystems can be considered being loosely coupled and having a high cohesion, each of them has an internal architecture, that at some level may become more highly coupled. For example the modules of the base software may use common global data. Typically these kinds of issues are resulting from the needs for performance optimization. In addition it is not common to have object-oriented design in this kind of embedded software, which in turn may result to a higher degree of coupling in the code.

The software is built of the source code level components by compiling and linking them to a single executable binary. Most of the components are developed by the organization but also some 3rd party components are used. The project is responsible of developing the base software and integrating it with the service and application components, which are partly developed in separate projects. The outcome of the project is a reference configuration which will be the baseline for the actual productization projects, including base software, service components and reduced subset of application components. Thus, the studied integrations include all types of the components used in the product.

The integration problems faced in case 6 were reported by the developers and captured in a class C CMMI appraisal for PI process area. The integration problems were also identified by analyzing the statistics from the build and linking phase. This was possible as the integration activities in the project were mainly related to the software construction phase, as described in [22].

A major problem for this organization is that changes to the source code are introduced without proper testing. Also late deliveries of functions are a problem as the functions often consists of sub-functions that are combined in the integration.

5.7. Case 7

The project in case 7 develops embedded software for a new generation of protection relays for electrical network. The development organization is small with around 15 developers. The main development team is located in two sites in two different time zones.

The product architecture resembles that of case 6, i.e. it consists of base software, application components and higher level services. The product has some common components with case 6, but it is based on other hardware. In addition some components, for example the base software, have a longer history, being released in the existing products. It was also well recognized by the development

team that partly due to its longer history, the base software had some highly coupled modules. Also, as in case 6, the performance optimization sometimes lead to the need for developing highly coupled modules inside the base software. The software is built of source code level components, either developed by the organization or 3rd party, compiled and linked to single executable binary. The project is responsible of further development of the base software and integrating it with the service and application components, which are partly developed in separate projects. The outcome of the project is a new product.

At the time of the study, the integrations were performed only with the components of base software and service components, while the application components were not yet included in the integration. The project size was over 500 KLOC, but when integrated with the application components, is expected to be >1000 KLOC.

The methodology used for finding and analyzing the reasons for the integration problems was same as for case 6, i.e. it was based on the class C CMMI appraisal and data collected during the build phase of the product's software.

For this case, three problems were reported by the integration and verification responsible persons. These are problems in the build environment, problems appearing in parts of the system that has not been changed, and that functions sometimes are untested when delivered for integration.

5.8. Conclusion of the case studies

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

The problems found in the product integration can be categorized into four different classes:

- Problems related to *inadequate selection and implementation of strategy*.
  If the wrong or no strategy is selected for the product integration process, the results will be that unnecessary work is done. Examples include:
  - Implicit strategy for the product integration results in uncoordinated deliveries from different subsystems making it difficult to perform meaningful integration tests
  - Missing strategy for error correction may result in changes that increase the risk for problems in product integration
- Problems related to *inadequate management of architecture and design.*
  This class concerns primarily the design and management of interfaces between different parts of the product or system, but also non-functional attributes of the product. Examples of problems are:
  - Architectural decisions are done without considering the full system, leading to discovery of problems related to non-functional attributes at integration time
  - Changes are made to interfaces without proper control. This leads to errors in the builds or initial integration testing
  - Changes in common resources (e.g. common include files) are not controlled. This results in errors appearing in other components which have not been changed
  - New functions are added and errors are corrected without proper investigation of consequences. The result may be new errors that influence the functionality and performance of the system more than the original problem
  - Errors appear in components which have not been changed due to changes in interfaces, i.e. changes are made in how two components interact, while also other components are using this interface
- Problems related to the *inadequate establishment or use of the integration environment.*
  The environment includes hardware, software, and test equipment needed for the integration. The following are examples from our case studies:
  - Problems appear as tests for the components are not run in the same type of environment as the integration test system. Different versions of hardware and test platform are used
  - The build environment is not prepared for new builds, e.g. results from earlier builds are not removed before a new generation of the system is started
  - Untested changes are introduced in the integration environment e.g. build scripts are changed without proper verification
- Problems related to *inadequate delivery of functions.*
  As the delivery for integration is based on a common understanding of priorities and needs from other parts of the system, problems in this class can cause considerable delays. Examples of problems:
  - Inconsistent code, i.e. functions that have only been partly implemented, is delivered for integration
  - Functions are not always delivered in time for integration or may be incompletely delivered. This leads to problems in the build process or in integration and system tests
  - Functions are not always fully tested when delivered for integration. This leads to problems in the build process or in integration and system tests

The proposed classification matches the different practices proposed for product integration and should be further refined as we learn more about the problems found in product integration.

One additional factor that may influence the results of our study is the company culture of the investigated companies. However, when comparing the performed practices and the problems found in the different cases, there is more resemblance between case 1 and 2 that are from different companies than between case 2 and 3 that both are from company B. In addition, except for case 6 and 7, the development organizations in company A are from different divisions, with different product development processes.

5.9. Problems in relation to reference models

The problems found in the cases are in this section related to the reference models. This gives an indication on how the reference models can help in avoiding the types of problems found. If a problem can be related to a practice that is not performed, but is described in the reference model, there is a possibility that the practice could help if implemented. However, if there are problems that we can relate to product integration, but cannot be related to the practices for a specific reference model, following that specific reference model does not support the organization in preventing that type of problem. The purpose of this analysis is thus to understand if there are problems in product integration that are possible to avoid if each of the reference models are expanded with the practices that are not explicitly described today.

For each reference model, a table summarizes the described product integration practices and the adherence to the tasks as observed in the cases. The seven columns describing the results from the case studies include: an indication for each case if the practice has been observed as performed (+), observed as not performed (−), not investigated or not possible to determine (?), and if there are indications of problems connected to the practice (indicated with the problem label). Only the explicitly described practices

have been included in the tables as this leaves less interpretation to the user of the reference model, and give more help.

Two issues limit the value of this analysis. The first is that explicit coverage of all the practices described in all reference models was not made in the data collection for all cases. This means that practices may be performed even if no evidence can be found in the material from the interviews and document reviews. This has been indicated with a questions mark (?) in the tables. The second issue is that problems may exist in the organization without indications in the table, based on the fact that not all practices were explicitly covered in the data collection for all case studies. However, all problems that have been captured have been possible to relate to practices that have been investigated.

5.10. ISO/IEC 12207:1995 compared to cases (Table 4)

The practices in ISO/IEC 12207:1995 cover 7 of the 17 unique problems found in the case studies. These are related to the practices used to ensure that the integrated software is ready for verification. This standard has no requirements on the handling of interfaces besides the definition of them, which represents the cause of many of the problems found in the case studies.

5.11. EIA-632 (Table 5)

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. 4 of the 17 unique problems have been related to practices described.

5.12. Capability maturity model integration (CMMI), version 1.1 (Table 6)

13 of the 17 problems encountered in the case studies regarding product integration can be related to practices that are described in the CMMI. Of the four that are not covered, three are related to late delivery of components to integration, and one is related to the strategy regarding error correction.

5.13. EAI-731.1 (Table 7)

As with CMMI, many problems found in the case studies can be related to practices in EIA-731.1. Of the 17 unique errors found in the cases, 13 are covered by the practices. The remaining four errors are related to the build environment and the criteria for integration.

5.14. ISO/IEC 15288, Systems engineering – system life cycle processes (Table 8)

The standard covers 11 of the 17 problems found in the case studies, and most of the other errors are related to the interface handling which is not explicitly covered.

Table 9 summarizes the use of practices derived from the reference models, and the problems identified in the case studies.

Table 4
ISO/IEC 12207:1995 compared to cases.

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
6. Define interfaces	–	–	–	+	–	–	–
10. Develop and document a set of tests for each requirement of the assembled components	+	–	+	–	+	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
12. Deliver/obtain components as agreed in the schedule	−1A	–	+	–	−5A	−6B	–
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–

Table 5
EIA-632 compared to cases.

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
6. Define interfaces	–	–	–	+	–	–	–
10. Develop and document a set of tests for each requirement of the assembled components	+	–	+	–	+	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–

Table 6
CMMI compared to cases.

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
3. Define and establish an environment for integration	+	+	+	+	−5B	+6C	+7B
4. Define criteria for delivery of components	−1B	−2A	+	–	+	–	–
6. Define interfaces	–	–	–	+	–	–	–
7. Review interface descriptions for completeness	−1C	−2C	–	−4A	-	+	+
8. Ensure coordination of interface changes	−1C	−2C	–	–	–	−6D	−7C
9. Review adherence to defined interfaces	–	–	+	+	–	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–

Through our case studies and the investigation of different reference models, we have found the following:

- Five of the practices (PI Practice 4, 7, 8, 11, and 12) indicate that problems may appear when the practice is not followed.
- In three instances, identified problem areas can be related to practices that are performed in the organizations. Two of these are related to the strategy definition (PI Practice 1), and are in fact referring to the lack of rules for corrections of errors. Hence, it may be that the integration strategy is available, but does not cover the rules for error corrections. The final problem is related to the build environment definition (PI practice 3) and is also an indication that the descriptions of practices do not cover the quality of the implementation.
- Table 10 illustrates also that none of the models include all necessary practices to avoid the problems found in the cases, e.g. for ISO/IEC 15288 we could associate 11 of the 17 problems found in the case studies to the product integration practices in that standard. The results confirm the need of a broader approach than is available in any of the examined reference models

Table 10
Number of errors for each case related to practices in the reference models.

	ISO/IEC 12207	EIA-632	CMMI	EIA-731.1	ISO/IEC 15288
Case 13 problems	2	1	2	3	2
Case 23 problems	0	0	2	2	1
Case 31 problem	1	1	1	1	1
Case 41 problem	0	0	1	1	0
Case 52 problems	1	0	1	1	2
Case 64 problems	2	1	3	3	3
Case 73 problems	1	1	3	2	2
Total17 problems	7	4	13	13	11

Our finding is that there is a small set of practices that need to be implemented to have working product integration. However, they are not sufficient, which is indicated by the larger set of practices described in different reference models. Further analysis shows that a combination of CMMI with either ISO/IEC

Table 7
EIA-731.1 compared to cases.

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
1. Define and document an integration strategy	?	+2B	+	+	+	?	?
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
6. Define interfaces	–	–	–	+	–	–	–
7. Review interface descriptions for completeness	−1C	−2C	–	−4A	–	+	+
8. Ensure coordination of interface changes	−1C	−2C	–	–	–	−6D	−7C
9. Review adherence to defined interfaces	–	–	+	+	–	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
12. Deliver/obtain components as agreed in the schedule	−1A	–	+	–	−5A	−6B	–
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–

Table 8
ISO/IEC 15288:2002 compared to cases

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
1. Define and document an integration strategy	?	+2B	+	+	+	?	?
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
3. Define and establish an environment for integration	+	+	+	+	−5B	+6C	+7B
5. Identify constraints from the integration strategy on design	?	?	?	?	?	?	?
9. Review adherence to defined interfaces	–	–	+	+	–	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
12. Deliver/obtain components as agreed in the schedule	−1A	–	+	–	−5A	−6B	–
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–
15. Record the integration information in an appropriate repository	–	–	–	–	–	–	–

Table 9
Performed activities and problems identified in the case studies.

Generic activity description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7
1. Define and document an integration strategy	?	+2B	+	+	+	?	?
2. Develop an integration plan based on the strategy	+	–	+	–	–	–	–
3. Define and establish an environment for integration	+	+	+	+	−5B	+6C	+7B
4. Define criteria for delivery of components	−1B	−2A	+	–	+	–	–
5. Identify constraints from the integration strategy on design	?	?	?	?	?	?	?
6. Define interfaces	–	–	–	+	–	–	–
7. Review interface descriptions for completeness	−1C	−2C	–	−4A	–	+	+
8. Ensure coordination of interface changes	−1C	−2C	–	–	–	−6D	−7C
9. Review adherence to defined interfaces	–	–	+	+	–	–	–
10. Develop and document a set of tests for each requirement of the assembled components	+	–	+	–	+	–	–
11. Verify completeness of components obtained for integration through checking criteria for delivery	−1B	–	−3A	+	–	−6A	−7A
12. Deliver/obtain components as agreed in the schedule	−1A	–	+	–	−5A	−6B	–
13. Integrate/assemble components as planned	+	+	+	+	+	–	–
14. Evaluate/test the assembled components	+	+	+	+	+	–	–
15. Record the integration information in an appropriate repository	–	–	–	–	–	–	–

15288:200 or EIA-733.1 would be sufficient to include all necessary practices to avoid the types of problems encountered in our case studies.

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

6. Discussion

The difficulties for product development organization found during integration are disrupting the progress of development projects and increase time-to-market. Problems origin for example in the lack of integration planning, insufficient management of interfaces, and inadequate preparation of components delivered for integration.

Our intent with this research has been to examine to which extent the practices described in reference models are useful as a support for development units. Five reference models have been analyzed. Practices as well as problems from seven development projects have been captured. We have based the investigation on the following questions and summarize here the results:

- How are the practices that are described in reference models useful for product development units aiming at improvements in the product integration process? The reference models can be used as tools for identifying weak areas in the product integration processes. Care must however be taken when selecting the reference model so that sufficient coverage is obtained.
- What is the core set of practices that can be identified to reduce problems in product integration? Through the case studies, five Product Integration practices have been identified to be necessary to perform to have successful product integration. These are PI 4 *Define criteria for delivery of components*, PI 7 *Review interface descriptions for completeness*, PI 8 *Ensure coordination of interfaces changes*, PI 11 *Verify completeness pf components obtained for integration through checking criteria for delivery*, and PI 12 *Deliver/obtain components as agreed in the schedule*. For the interface handling, also PI 6 *Define interfaces* is important as PI 7 relies on that practice. The same reasoning can be applied on PI 2 *Develop an integration plan based on the strategy* which is a prerequisite for PI 12.
- Is it appropriate to combine reference models to provide better support to product development units, and how can this be done? The analysis of existing reference models shows that none of the investigated models cover the problem situations for the investigated product development organizations regarding product integration. This leads to our conclusion that a combination of the content in the reference models can be helpful for development organizations when designing and improving the product integration process.

Our suggestion to companies that would like to improve the product integration processes is to use the set of 15 practices described in Section 4.6 and perform an assessment on the current practices. In addition to this, the problem areas found should together with the assessment results be the basis for any improvement effort.

One additional conclusion is that a continued development towards an agreed body-of-knowledge for the product integration area is needed. This can be achieved through consolidation and further validation of existing reference models. Finally, as a result of our studies, we see the need to perform additional investigations to understand 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. Additional organizations using different technologies should be investigated and compared to clarify if there are dependencies between the type of application and the needed practices. A related direction is to look at the influence architectural decisions have on product integration. Also, 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 reference models investigated in this article 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.

Acknowledgements

We would like to thank all participants in the case studies for their time and patience. This work has been partially supported by the KK-foundation (KKS) in Sweden through the SAVE-IT project.

References

[1] ANSI, American National Standards Institute, <http://www.ansi.org/>, 2007.
[2] ANSI/EIA-632-1999, Processes for engineering a system, Electronic Industries Alliance, Government Electronic and Information Technology Association, 1999.
[3] M. Bajec, D. Vavpoti, M. Krisper, Practice-driven approach for creating project-specific software development methods, Information and Software Technology 49 (2007) 345.
[4] J. Campanella, Principles of Quality Costs: Principles Implementation and Use, ASQ Press, Milwaukee, WN, USA, 1999.
[5] R.I. Chiang, V.S. Mookerjee, A fault threshold policy to manage software development projects, Information System Research 15 (2004) 3–21.
[6] C.G. Chittister, Y.Y. Haimes, Systems integration via software risk management systems, man and cybernetics, part A, IEEE Transactions on 26 (1996) 521–532.
[7] F.A. Cummins, Enterprise Integration: An Architecture for Enterprise Application and Systems Integration, John Wiley & Sons, 2002.
[8] M. de Jonge, Package-based software development, in: Euromicro Conference, 2003, Proceedings 29th, 2003, pp. 76–85.
[9] G.R. Djavanshir, R. Khorramshahgol, Key Process Areas in Systems Integration, in: IT Professional, vol. 9, 2007, pp. 24–27.
[10] A.H. Dogru, M.M. Tanik, A process model for component-oriented software engineering, IEEE Software 20 (2003) 34–41.
[11] EIA-731.1, Systems engineering capability model, Electronic Industries Alliance, 2002.
[12] F. Ekdahl, S. Larsson, Experience report: using Internal CMMI appraisals to institutionalize software development performance improvement, in: 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 06), 2006, pp. 216–223.
[13] B. Fitzgerald, An empirical investigation into the adoption of systems development methodologies, Information and Management 34 (1998) 317–328.
[14] M. Fowler, Continuous Integration, <http://www.martinfowler.com/articles/continuousIntegration.html>, (2006).
[15] D. Garlan, Software architecture: a roadmap, in: Proceedings of the Conference on the Future of Software Engineering, ACM Press, Limerick, Ireland, 2000, pp. 91–101.
[16] I. Gorton, Essential Software Architecture, Springer, 2006.
[17] D. Houston, An experience in facilitating process improvement with an integration problem reporting process simulation, Software Process: Improvement and Practice 11 (2006) 361–371.
[18] IEEE1220-2005, IEEE standard for application and management of the systems engineering process, Institute of Electrical and Electronics Engineers, 2005.
[19] IEEE, The Institute of Electrical and Electronics Engineers, <http://www.ieee.org/>, 2007.
[20] INCOSE, International Counsil on Systems Engineering, <http://www.incose.org/>, 2007.
[21] ISO9001:2000, Quality management systems – Requirements, ISO, 2000.

[22] ISO, International Standardization Organization, <http://www.iso.org>, 2007.
[23] ISO/IEC12207:1995, Information technology – Software life cycle processes, ISO/IEC, 1995.
[24] ISO/IEC15288:2002, Systems engineering – Systems life cycle processes, ISO/IEC, 2002.
[25] G. Karsai, J. Sztipanovits, A. Ledeczi, T. Bapty, Model-integrated development of embedded software, Proceedings of the IEEE 91 (2003) 145–164.
[26] D.R. Kuhn, On the effective use of software standards in systems integration, in: Systems Integration '90, Proceedings of the First International Conference on (1990) pp. 455–461.
[27] S. Larsson, Improving software product integration, Dept. of Computer Science and Electronics Mälardalen University, 2005.
[28] S. Larsson, I. Crnkovic, Case Study: Software Product Integration Practices, in: 6th international conference Profes, June, 2005, Oulu Finland, 2005, pp. 272–285.
[29] S. Larsson, I. Crnkovic, F. Ekdahl, On the expected synergies between component-based software engineering and best practices in product integration, in: Proceedings – 30th EUROMICRO Conference, Aug 31-Sep 3 2004, vol. 30 (IEEE Computer Society, Los Alamitos; Massey University, Palmerston, CA 90720-1314, United States;New Zealand, Rennes, France, 2004) pp. 430–436.
[30] S. Larsson, P. Myllyperkiö, F. Ekdahl, Product Integration Improvement Based on Analysis of Build Statistics, in: ESEC/FSE, Dubrovnik, Croatia, 2007.
[31] M. Leszak, D.E. Perry, D. Stoll, Classification and evaluation of defects in a project retrospective, Journal of Systems and Software 61 (2002) 173–187.
[32] D.S. Linthicum, Enterprise Application Integration, Addison-Wesley, 1999.
[33] E.G. Nilsson, E.K. Nordhagen, G. Oftedal, Aspects of systems integration, in: Systems Integration '90, Proceedings of the First International Conference on (1990) pp. 434–443.
[34] RTI, The economic impacts of inadequate infrastructure for software testing, in: National Institute of Standards and Technology, Gaithersburg, MD, USA, 2002.
[35] W.A. Ruh, F.X. Maginnis, W.J. Brown, Enterprise Application Integration, Addison-Wesley, 2000.
[36] A.P. Sage, L. Charles, Lynch, Systems integration and architecting: an overview of principles, practices, and perspectives, Systems Engineering 1 (1998) 176–227.
[37] M. Schulte, Model-based integration of reusable component-based avionics systems – a case study, 2005, pp. 62–71.
[38] SEI, Appraisal Requirements for CMMI, Version 1.1 (ARC, V1.1), Carnegie Mellon University, Software Engineering Institute, 2001.
[39] SEI, CMMI® for Development, Version 1.2., Pittsburgh, PA, USA, 2006.
[40] SEI, Software Engineering Institute, <http://www.sei.cmu.edu/>, 2007.
[41] V. Stavridou, Integration in software intensive systems, Journal of Systems and Software 48 (1999) 91–104.
[42] V. Stavridou, Integration standards for critical software intensive systems, in: Software Engineering Standards Symposium and Forum, 1997. 'Emerging International Standards'. ISESS 97, Third IEEE International (1997) pp. 99–109.
[43] R.K. Yin, Case Study Research: Design and Methods, Sage Publications, 2003.