Process Patterns for Software Systems In-house Integration and Merge – Experiences from Industry

Rikard Land, Ivica Crnković, Stig Larsson
Mälardalen University, Department of Computer Science and Electronics
PO Box 883, SE-721 23 Västerås, Sweden
+46 21 10 70 35
{rikard.land, ivica.crnkovic, stig.larsson}@mdh.se, http://www.idt.mdh.se/{~rld, ~icc}

Abstract

When an organization faces new types of collaboration, for example after a company merger, there is a need to integrate the existing software. Two main process challenges are how to arrive at a realistic vision of a future integrated system, and how to actually carry out the integration process.

We have performed a multiple case study, consisting of 9 cases. This paper presents the observations made in the form of recurring patterns that can be used as recommendations for other organizations facing the same challenge. Also discussed are the similarities and differences between already known software process best practices and the integration patterns found.

1. Introduction

From time to time within an organization, two or more in-house developed software systems address similar needs, and there is an overlap in functionality. This typically happens when the organization changes through new types of collaborations and mergers. The software may be the core products of the companies, or some support systems for the core business. If the software systems are mainly used in-house, performing further evolution and maintenance of two systems in parallel seems unfeasible. If the software systems are products of the company, it makes little sense to offer customers two similar products. In either case, the organization would ideally want to take the best out of the existing systems and integrate them with as little effort as possible. This could for example mean reusing components of the systems in a new system, integrate them more loosely, discontinuing one system and extending the other, or even discontinuing both and start development of a new generation. In practice many different decisions are made, not necessarily the optimal ones. Our goal is to identify the main characteristics of this process: Which are the driving forces, the most important system characteristics, and what rationale lies behind the decision for the changes and the changes process? To investigate this, we have carried out a multiple case study [14] with 9 cases from 6 organizations that have gone through such an integration process. In the present paper, we have chosen to report the experiences in the form of recurring patterns.

For an organization that has identified a functional overlap and a need for integration, two main challenges are: 1) how to develop a vision for a future, integrated system, and 2) how to reach there within reasonable time, using reasonable resources. See Figure 1.

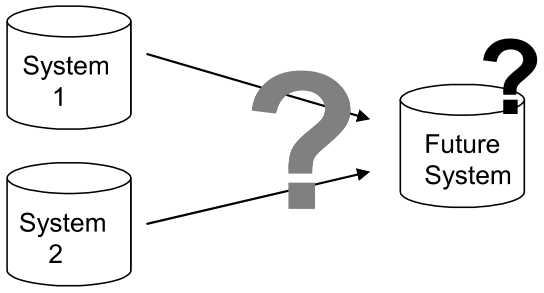

Figure 1: The two challenges – the future system and the path there.

These two challenges can be visualized as two processes: a vision process and an actual integration process. Although one might initially think of them as sequential – first define a vision, then implement it – we prefer to visualize them as two processes carried out iteratively or in parallel, each affecting the other. See Figure 2. This is in line with the results from the cases.

The questions addressed by this paper are:

Q1. Which are common experiences (good and bad) concerning these processes?

Q2. To what extent are the lessons learned from these experiences possible to generalize into recommendations for other organizations?

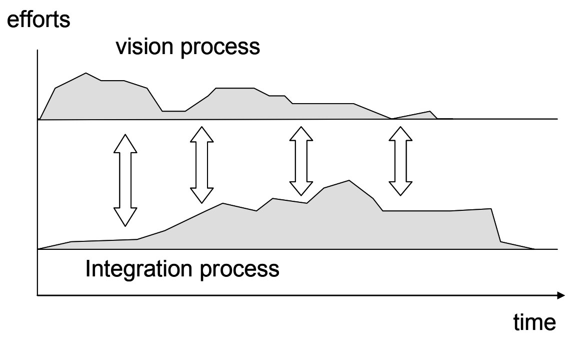

Figure 2: Interaction between vision and integration process.

Q3. How do the experiences from integration relate to other known good practices in other development activities?

Section 2 describes related work, section 3 describes the methodology used in the research, and section 4 introduces the cases. Section 0 answers Q1 and Q2 by presenting experiences from the cases in the form of patterns, and is subdivided into the vision process (section 5.1), the integration process (section 5.2), the interaction between these (section 5.3), and some observations about the distributed nature of the organizations (section 5.4). Section 6 concludes the paper by summarizing the most important observations, discussing answers to 0, and outlining future work.

2. Related Work

Although there is much published experience on software integration, most of it concerns slightly different types of integration. Three major fields of software integration are component-based software [13], open systems [10], and Enterprise Application Integration, EAI [5,11]. In a previous survey of existing approaches to software integration [7], we found that there is basically no existing literature that directly addresses the context of the present research: integration of software *completely controlled and owned within an organization*.

The context of the present research in practice means that there are two different software organizations that need to start cooperating, which most often includes the known problems of distributed software development: cultural differences (including "company cultures"), a "we vs. them" attitude, and in many cases also different languages and cooperation across different time zones [2,3,6].

In the overall process we have observed a clear distinction between a *vision process* and the *integration process.* This observation can be compared with distinguishing a decision process and a development process [1,4,12]. In these related works the discussions are focused on relations between these processes and conveying messages between the stakeholders in these processes. In our case these relations will be specific as the vision and the decisions are based on two or more existing systems and according to that two or more groups of stakeholders.

3. Research Methodology

The multiple case study [14] consists of nine cases from six organizations that have gone through an integration process. Our main data source has been interviews, but in some cases we also had access to certain documentation. The interviewees were asked about the history of their systems, the architecture of the existing and integrated system, the reasons for integrating, and what they would do different next time. The open-ended questions were focused around architecture and processes, and the copied out interview notes were sent back to the interviewees for feedback and approval. In one case (F1) one of the authors (R.L.) also participated as an active member. Details regarding the research design and the material from the interviews (the questions and all answers) are available in a technical report [9].

4. The Cases

The cases come from different types and sizes of organizations operating in different domains, the size of the systems range from to a maintenance and development staff of a few people to several hundred people, and the types of qualities required are very different depending on the system domain. What the cases have in common though is that the systems have a significant history of development and maintenance.

The cases are summarized in Table 1. They are labeled A, B, etc. Cases E1, E2, F1, F2, and F3 occurred within the same organizations (E and F). For the data sources, the acronyms used are I_X for interviews, D_X for documents, and P_X for participation, where X is the case name (as e.g. in I_A, the interview of case A), plus an optional lower case letter when several sources exist for a case (as e.g. for interview I_{Da}, one of the interviews for case D). I_X:n refers to the answer to question n in interview I_X. In the present paper, we have provided explicit pointers into this source of data.

Table 1: Summary of the cases.

	Organization	System Domain	Goal	Information Resources
A	Newly merged international company	Safety-critical systems with embedded software	New HMI[*] platform to be used for many products	*Interview:* project leader for "next generation" development project (I_A)
B	Organization within large international enterprise	Administration of stock keeping	Rationalizing two systems within corporation with similar purpose	*Interview:* experienced manager and developer (I_B)
C	Newly merged international company	Safety-critical systems with embedded software	Rationalizing two core products into one	*Interviews:* leader for a small group evaluating integration alternatives (I_{Ca}); main architect of one of the systems (I_{Cb})
D	Newly merged international company	Off-line management of power distribution systems	Reusing HMI[*] for Data-Intensive Server	*Interviews:* architects/developers (I_{Da}, I_{Db}).
E1	Cooperation defense research institute and industry	Off-line physics simulation	Creating next generation simulation models from today's	*Interview:* project leader and main interface developer (I_{E1}) *Document:* protocol from startup meeting (D_{E1})
E2	Different parts of Swedish defense	Off-line physics simulation	Possible rationalization of three simulation systems with similar purpose	*Interview:* project leader and developer (I_{E2}) *Documents:* evaluation of existing simulation systems (D_{E2a}); other documentation (D_{E2b}, D_{E2c}, D_{E2d}, D_{E2e}, D_{E2f})
F1	Newly merged international company	Managing off-line physics simulations	Possible rationalization by using one single system	*Participation:* 2002 (R.L.) (P_{F1a}); currently (R.L.) (P_{F1b}). *Interviews:* architects/developers (I_{F1a}, I_{F1b}); QA responsible (I_{F1c}) *Documentation:* research papers (D_{F1a}); project documentation (D_{F1b})
F2	Newly merged international company	Off-line physics simulation	Improving the current state at two sites	*Interviews:* software engineers (I_{F2a}, I_{F2b}, I_{F2f}); project manager (I_{F2c}); physics experts (I_{F2d}, I_{F2e})
F3	Newly merged international company	Software issue reporting	Possible rationalization by using one single system	*Interview:* project leader and main implementer (I_{F3}) *Documentation:* miscellaneous related (D_{F3a}, D_{F3b})

Some cases have successfully performed some integration, others are underway. All cases reported both successes and mistakes, which are all taken into account in the present paper.

5. Analysis

This section presents recurring characteristics of the processes in the cases.

5.1 The Vision Process

The obvious starting point for integration is often an initial vision from higher management (I_{Ca}:6, I_{Cb}:6, I_{Db}:3,5,6, I_{F2c}:3). The goal is to rationalize the activities related to the products (maintenance, data overlap, duplicated processes) (I_A:2,3, I_B:1, I_{Ca}:6, I_{Cb}:6, I_{Db}:3,5,6, P_{F1a}, P_{F1b}, D_{F1b}, I_{F2d}:3). In cases C and F1, higher management gave directions how a system merge should be achieved: "try to agree and reuse as much as possible" (I_{Cb}:6, also P_{F1a}). In case C, this

[*] HMI=Human-Machine Interface

caused an expensive delay as well as other problems (I_{Ca}:6,7), and in case F1 the architecture and outlined integration plan felt watered-down (D_{F1a}, I_{F1c}:6), and nothing happened to realize it (P_{F1a}, P_{F1b}). Case E1 may be mentioned as a counter-case, where enthusiasm and a successful combination of people, and an eagerness to get started overcame many obstacles and seem to have been the starting point (I_{E1}:6,7,9,11); at the startup meeting, the interviewee already had the fundamental structure clear (I_{E1}:7, D_{E1}).

We have observed the following seven patterns of the vision process:

Small evaluation group. Statement: *After higher management has identified some potential benefits with integration, a small group of experts should be assigned to evaluate the existing systems from many points of view and describe alternative high-level strategies for the integration*. In cases C and F1 a small group evaluated the existing systems with the specific goal to identify how integration should or could be carried out, at the technical level (I_{Ca}:6, I_{Cb}:6, I_{F1c}:6, P_{F1a}, P_{F1b}, D_{E1a}). In case F1, users were also involved in this process, in order for them to grade different features of the existing systems (P_{F1a}, D_{F1a}). It is important to involve both sides, as no single individual has overview of all systems (both cases concern newly merged companies). Also, everyone involved is partial and there is a clear risk that everyone "defends" their own system (I_{Cb}:6), there must be an open mind for other solutions than "ours" (I_{F3}:11). In the cases it appears that there has indeed been a good working climate with a "good will" from everyone (I_{Cb}:6, P_{F1a}). In both cases this was considered a good scheme; in case C the architects immediately saw that there were no major technical advantages of either system, and wanted to immediately discontinue one of the two systems, indifferent which, rather than trying the (I_{Cb}:6). The late decision (indeed, to discontinue one of the systems) was due to other reasons (see "timely decisions**"** below. A similar scheme was used in case E2, an external investigation was made, however with less technical expertise (I_{E2}:6, D_{E2a}).

Life cycle phase of existing systems. Statement: *The life cycle phases of the existing systems affect the choice of integration strategy* (I_{Cb}:1,7, I_{Cb}:6, I_{E1}:4, I_{F2e}:6, I_{F2a}:3). For example, proven high-quality systems are not easily discarded, and systems considered aged are candidates for retirements. In case C, a new generation of both systems was being developed, but not yet released, and the obvious choice would seem to be to discard either of them before release (I_{Cb}:7, I_{Cb}:6); however development did continue until both systems were released, which led to lots of extra costs and problems (I_{Cb}:6). In case D, one of the existing HMIs was considered aged and was replaced be another (I_{Db}:3). In case F2, one of the sites was about to develop something new, while the other had realized some fundamental problems with the physical models their software embedded (I_{F2e}:6, I_{F2a}:3). This led to a successful common development project, however suffering from a lack of resources (see "commitment" pattern in section 5.2).

Reusing experience from existing systems. Statement: *All experience of the existing systems, in terms of e.g. user satisfaction and ease of maintenance must be collected in order to be able to describe the envisioned system properly* (I_A:6, P_{F1a}, D_{F1a}, I_{F2e}:6, I_{F2f}:6, I_{F3}:11). Ideally, one would like to define the new system as consisting of the best parts of the existing systems; however, this is in practice not as simple as it first may seem. The requirements on the future system are clearly dependent on the experience of the previous systems, and can be stated in terms of existing systems (I_A:6, P_{F1a}, D_{F1a}, I_{F3}:6). However, this means that the requirements need not (some of the sources even say should not) be too detailed (I_A:5,6,11, I_{C1a}:6, P_{F1a}, D_{F1a}). In case A, the development organization explicitly asked sales people for "killing arguments" only, not a detailed list of requirements (I_A:5). This, combined with the experience and understanding of the existing systems, makes a detailed list of requirements superfluous (i.e. during these early activities; later a formal requirements specification may be required). The people developing the vision of the future system (e.g. a small evaluation group) need to study the other systems, preferably live (I_{Ca}:6, D_{E2a}, I_{F3}:6). Case F2 involves complex scientific physics calculations, and the study of the existing systems' documentation of the implemented models was an important activity (I_{F2e}:6, I_{F2f}:6). When looking at the state of the existing systems, an open mind for other solutions than the current way of doing things is essential (I_{F3}:11). Reuse of experiences in the cases, divided into requirements and architectural solutions is elaborated elsewhere [8].

Improve the current state. Statement: *To gain acceptance, the efforts invested in the integrated system must not only present the same features as the existing system, but also improve the current state*. The existing systems must be taken into account (see pattern "reusing experience from existing systems"), but one should not be restricted by the current state (I_{F2f}:6); in case F2, it was indeed considered a mistake to keep the old data format and adapt new development to it (I_{F2a}:9, I_{F2d}:7,9,11). The actual needs must be more important than to preserve the features of the existing systems (I_{F3}:11). One interviewee stated that a new system would take ~10 years to implement, and a merged (and improved) system must be allowed to take some years as well (I_{F2f}:6). In case E1, integrating several small, separate pieces as was envisioned required a more structured language (Ada), even though it would in principle be possible to reuse many existing parts as they were written in Fortran (I_{E1}:6); the organization was interested in Ada as such, which also contributed to this choice (I_{E1}:7).

Timely decisions. Statement: *Making decisions in a timely manner is important* (I_{Ca}:6, I_{Cb}:6,11). When no decisive technical information has been found, a decision should be made anyway. In case C, the decision to discontinue one of the systems could have been made much earlier, as no new important information surfaced during the endless meetings with the small technical group (I_{Cb}:6). This means that one year of development money was wasted on parallel development, and the discontinued system has to be supported for years to come (I_{Ca}:6, I_{Cb}:6). “It is more important with a clear decision than a ‘totally right’ decision” (I_{Cb}:11). You cannot delegate the responsibility to agree to the grassroots (I_{Cb}:6). “Higher management must provide clear information and directives… It is… unproductive to live in a long period of not knowing” (I_{Cb}:11).

Sufficient analysis. Statement: *Before committing to a vision, sufficient analysis must be made.* Obvious as that may seem, the difficulty is the tradeoff between the need for understanding the existing systems well enough without spending too much time. In case F2, insufficient analysis caused large problems: what was believed to involve only minor modifications resulted in complete re-design and implementation (I_{F2a}:9, I_{F2b}:9, I_{F2c}:3, I_{F2d}:6, 11). One method of ensuring sufficient analysis could be to use the “small evaluation group” pattern. Of course, pre-decision analysis somewhat contradicts the pattern “timely decisions”; a stricter separation from the actual integration process is also introduced, implying a more waterfall-like model which might not be suitable (I_{F1b}:5,6).

Consider commercial alternatives. Statement: *When the existing systems do not embed core knowledge about the domain of the organization, the best alternative may be to choose an existing system.* There might e.g. be commercial or open source alternatives. This actually happened in one of the cases, where software issue tracking systems had been developed in-house, but after the company merger a new, commercial system was acquired and implemented throughout the organization (I_{F3}:6, 7).

In addition to these seven patterns, two more observations should be described:

No vision and no integration. Some interviewees proposed the opinion of not integrating at all. “Why integrate at all?” (I_{Cb}:7) is indeed a valid question, which will arise if a decision is not accompanied with priority and enough resources (I_{F1b}:3, I_{F1c}:6,9,11, P_{F1a}). Sometimes it might simply not be worth the effort to integrate – will the future savings through rationalization be larger than the integration efforts? (I_{F1c}:9, I_{F2d}:3). Reasonable project plans for reaching the vision must be considered; in case E2 there were very few resources available, which led to a very modest vision, in practice meaning no integration (I_{E2}:6). This is further discussed in section 5.3 concerning the interaction between the vision process and the integration process.

Architecture. Essential when developing a vision for a future system are the architectures of the existing systems, and the technologies used. Are they compatible or not (I_{F2a}:1, I_{F2b}:1,7)? Will it involve more effort to merge than to discontinue one and evolve the other system (I_{Cb}:6)? We believe involving architecture in the process is so important that it deserves a separate paper and will not elaborate it further here [8].

5.2 The Integration Process

Recalling Figure 1 again, the desire is to make the systems converge. However, without interference the development will rather diverge. Convergence must be forced; without such forces, the vision will never be reached (I_{Da}:3, , I_{Db}:3,5,6, I_{F1c}:6,9, P_{F1a}, P_{F1b}). Obvious as that may seem, several of the cases have had large problems with this. In several cases, the decision about the future was not accompanied by any (or at least enough) concrete measures to achieve integration (I_{Da}:3, I_{Db}:3,5,6, I_{F1c}:6,9, P_{F1a}, P_{F1b}).

In the cases, the following six patterns were found concerning the integration process:

Strong project management. Statement: *To run integration efforts in parallel with other development efforts, a strong project management is needed* (e.g. I_{F1c}:9,11, I_{F2b}:5,11, I_{F2e}:9,11). To be able to control development, higher management and project management must have economical means of control (I_{Ca}:11, I_{F1b}:11). In case C, not until economical means of control were put into place did development of the system-to-be-discontinued stop (I_{Ca}:6). Case E1, a cooperation led by a research institute, can serve as a counter-example. Here, enthusiasm apparently was the driving force, and the lack of strict management was even pointed out as contributing to success (I_{E1}:9,11). Although we agree it is important to create a good and creative team spirit, we believe it would be bad advice to recommend weak or informal project management, at least for larger projects.

Commitment. Statement: *It is not possible to succeed with integration if the efforts are half-hearted.* Commitment is needed from all stakeholders (I_{F1b}:11, I_{F1c}:11), which must also be accompanied with enough resources (I_{F1c}:11). In case F2 it was pointed out (based on negative experience) that for strategic work as integration is, one cannot assign just anyone with some of the required skills; the right (i.e. the best) people must be assigned, which is a task for project management (I_A:11, I_{F2b}:11, I_{F2d}:9,11, I_{F2e}:9,11). Realistic plans must be prepared, and resources assigned in line with those plans (I_{F1c}:11). When directives and visions are not accompanied with resources, integration will be fundamentally questioned (I_{F1b}:3, I_{F1c}:6,9). When there is a lack of resources, short-term goals tend to occupy the mind of the people involved. Without a minimum effort in integration, the environment and the vision will change more rapidly than the integration makes progress, which means only a waste of resources. Integration will be doubted, which takes even more energy from the people involved. A long period of integration is problematic, since you need to maintain the existing system meanwhile (and for a while after they are retired as well) (I_{F2f}:6).

Cooperative grassroots. Statement: *In order to succeed, the "grassroots" (i.e. the people who will actually do the hard work) must be cooperative, both with management and each other.* The overall goals must be clear and they need to get commitment and "buy-in" (I_{F1b}:11). The organization must be kept motivated (I_{Cb}:11). In case D, the grassroots considered explicitly whether cooperation was of benefit to themselves (I_{Db}:6); they decided that for cooperation to succeed they needed to show they were willing to build trust, that they had no hidden agenda (I_{Db}:6,11). The "not invented here syndrome" is dangerous for cooperation (I_{Db}:6, I_{F1c}:11). Case E1 illustrates that a fun project with fun people may drive the integration so that the need for strict management project schedules is reduced (I_{E1}:9); what contributed most to success were the fun people and the lack of strict management (I_{E1}:11).

Make agreements and keep them. Statement: *To be able to manage and control a distributed organization formal agreements must be made and honored.* In case F2, it was pointed out as a big problem that requirements and design evolved driven by implementation (I_{F2b}:6, I_{F2c}:9, I_{F2d}:6, 11). Even in the informally managed case E1, the importance of agreeing on interface specifications and keeping them stable was emphasized (I_{E1}:7,9). More formalism than usual is required, you must have agreements written down and then stick to them (I_{F1c}:9,11).

Common development environment. Statement: *To be able to cooperate efficiently, a common development environment is needed* (P_{F1b}, I_{F2b}:6,11, I_{F2e}:11,12, I_{F2f}:12). With "development environment" we include e.g. development tools, platforms and version control systems. In case F2, it was difficult to synchronize the efforts (I_{F2e}:11); e.g. source code was sent via email and merged manually (I_{F2b}:6). In case F1, the difficulties of accessing the other site's repository caused an unnecessarily long period of (unknowing) parallel development (P_{F2b}).

Achieving momentum. Statement: *Achieving "momentum", i.e. an inner driving force is desirable.* (I_{F2f}:9) The external converging forces cannot be too strong for too long, which would take a lot of energy from the staff and the organization, will create stress and tension, and may also lead to a recurring questioning about the purpose of integration (I_{F1b}:3,11, I_{F1c}:6,9). One of the interviewees in case F1 (which has not made significant measurable progress during the 4 years that have passed since the company merger) asked "from where comes the driving force?" (I_{F1c}:9), pointing at the fact that integration is not a goal in itself. (These terms: converge, diverge, driving force, momentum, were terms used by many of the interviewees themselves).

5.3 The Interaction between Vision and Integration Processes

We made the following observations concerning the interaction between vision and integration process:

Stepwise delivery. Typically, the vision lies far into the future, and integration processes are less predictable than other development projects (I_{F2c}:10,12). Maintaining the long-term focus without some way to monitor and measure progress is impossible (I_A:6,9, I_B:1, I_{Da}:12, I_{Db}:6, I_{F1b}:6, I_{F2c}:6,11, I_{F2f}:6). In contrast to development of new products, or new product versions, these activities are performed in parallel and often not considered the most important. For these reasons the decisions regarding the integration process do not only depend on the process itself, but also on many unrelated and unpredictable reasons. Stepwise deliveries and prototyping have been used for new development to increase process flexibility and this was also a recurring opinion among the interviewees. This could be one way of achieving the desirable momentum. There were some variations on this theme:

- Some of the interviewees maintained that there must be a focus on deliveries that gives user value, and a clearly identified customer (I_B:1,7,11,13, I_{F1b}:6,11). If it is possible to utilize a customer delivery to perform some of the integration activities, this will be the spark needed to raise the priority, mobilize resources, gaining commitment etc. (I_{F2c}:6,11). However, it should also be noted that customer delivery projects typically have higher priority than long-term goals such as integration, and may steal resources and commitment from the integration process. The extreme would be to focus only on immediate needs, questioning the need of integration at all (I_{F1b}:3,11, I_{F1c}:6,9).

- Case A used prototyping as a way to show an early proof of concept (I_A:1,6,9,11).
- In some cases where it has been difficult to formulate, or agree on, or commit to a vision, the opinion has been raised that you rather need to move on and do something more concrete. There might be too many unknowns, and the best way to carve out a more concrete vision is to do something that is useful in the shorter term, and use it as a learning experience (I_{F2c}:11, I_{F2f}:6). In case F2 requirements and design evolved uncontrolled, driven by implementation (I_{F2b}:6, I_{F2c}:9, I_{F2d}:6,11); it would have been better to either freeze the requirements or to include constant change into the development model.
- For a large system, a waterfall model is not suitable (I_{F1b}:5,6). It is often considered too risky to define the complete integrated system and implement it, as this runs the risk of not being feasible at time of delivery; there is a too long time to return of investment (I_B:1). Closely associated is the approach of a loosely integrated system: an integration point should be found and all subsequent activities, although run as separate delivery projects, will little by little make integration happen (I_B:6,7, I_{F1b}:6,7,8,11; the proposed integration point in case F1 was a data storage format). There is however a tradeoff to be made, there are typically some common fundaments that need to be built first (P_{F1a}, D_{F1a}, I_{F2e}:7).
- In order to develop and install a number of customer-specific systems in parallel, divergence can be allowed, if there are mechanisms that will enforce standardization and convergence from time to time (I_B:7,11,13).

Relation to other development activities. As integration has to be done in parallel with the ordinary work within the organization, this often leads in another direction (I_{F1a}:9). There is a need to synchronize all parallel development efforts within the company, otherwise projects run too freely and sub-optimal solutions are created (I_{F1c}:6). This additional complexity is not explicitly addressed by the present research.

5.4 Distributed Software Development

All cases involve distributed organizations, as the need for software integration typically comes from new collaborations such as company mergers. The well-known problems of alleviating distance (both physical and cultural) [2,3,6] were discernible in the cases. Many of the interviewees emphasized the need for meeting in person, and also the benefits of job rotation (I_A:11, I_{F2b}:11, I_{F2c}:9,11, I_{F1a}:11, I_{F1c}:11, I_{F2d}:9, I_{F2e}:11).

Two particular "software culture" differences were described:

- In the small evaluation group of case C, the US organization sent managers while the Swedish organization wanted technicians to evaluate the systems (I_{Ca}:6).
- In case A, the Swedish site had a like for a particular commercial platform, while the German counterpart strongly advocated open source (I_A:2,7,8).

In cases A and C, the difficulties with distributed development were solved by assigning the task to one single site. In case A, development was assigned to the site historically strongest in HMIs (I_A:1), and in case C, discontinuing one system in practice meant a possibility to reduce other development sites (I_{Cb}:6). Local development is, if possible, the preferred way of addressing the problems with distributed development. However, this assumes there are indeed enough resources available at that site (which may not always be the case). This also assumes that there is enough knowledge at that site, which might not be possible if a tighter merge of the software is chosen than in the cases A (involving new development) and C (where one system was discontinued).

6. Summary and Conclusions

We have studied 9 cases of software integration, to answer the three questions that were asked in the beginning.

6.1 Summary of Patterns

To answer the first question, Q1 (about common experiences) we chose to formulate the results in terms of recurring *patterns*. We have distinguished two processes: A vision process and an integration process. Seven patterns were found in the vision process: *small evaluation group*; *life cycle phase of the existing systems* (may be beneficial); *reusing experience from existing systems*; *improve the current state*; *timely decisions*; *sufficient analysis*; *consider commercial alternatives* (the last is only appropriate if the systems are not core products). Six patterns were found concerning the integration process: *strong project management*; *commitment*; *cooperative grassroots*; *make agreements and keep them*; *common development environment*; *achieve momentum*. The main pattern found concerning the interaction between the vision and integration processes was *stepwise delivery*. This is the preferred way to monitor progress, maintain financing and commitment. It also provides a possibility to learn and refine the vision along the way.

The fact that the same patterns replicated themselves across the heterogeneous systems and organizations of the cases gives some confidence about the generality of the results, thus addressing Q2 (the possibility to generalize these experiences).

6.2 What Makes Integration Specific?

We can recognize many of the presented patterns being proposed as important factors in development software processes, or as important activities from experiences from best practices. However, the fact that the organizations of the cases found difficulties implementing the patterns indicates that during integration some already known "best practices" [4] must be implemented even stronger than usual, or may require much more efforts, or are different in technical details.

We believe that the higher importance of some patterns can be explained by other factors than the specific integration context. The patterns *strong project management*; *commitment*; *cooperative grassroots*; *make agreements and keep them*; *common development environment* seem to come mainly from the fact that there are two (distributed) groups involved; these patterns are thus recommendable to every distributed software development effort. The importance of some other patterns can be explained with the long time scale of integration and large organizations: *improve the current state*; *stepwise delivery*. Part of what makes integration unique is that these difficulties occur simultaneously. Some of the patterns seem to be more specific to the integration context, with two existing systems (not only one, as during ordinary evolution) and two groups of people with no one having complete overview or knowledge of both systems: *small evaluation group*; *reusing experience from existing systems*; *life cycle phase of the existing systems*.

6.3 Future Work

Answering the third question (which experiences are specific to integration) in more depth could involve refining the experiences into a process (model) for integration.

The two interrelated processes described in the present paper do not explain the complete course of events in the cases. For example, the architecture of the existing systems also heavily influences what is possible to do. We are currently viewing the same material from this point of view as well [8].

7. Acknowledgements

We would like to thank all interviewees and their organizations for sharing their experiences and allowing us to publish them.

8. References

[1] Bonner J. M., Ruekert R. W., and Walker O. C., "Upper management control of new product development projects and project performance", In *Journal of Product Innovation Management*, volume 19, issue 3, pp. 233-245, 2002.

[2] Carmel E., *Global Software Teams - Collaborating Across Borders and Time Zones*, ISBN 0-13-924218-X, Prentice-Hall, 1999.

[3] Carmel E. and Agarwal R., "Tactical Approaches for Alleviating Distance in Global Software Development", In *IEEE Software*, volume 18, issue 2, pp. 22-29, 2001.

[4] CMMI Product Team, *Capability Maturity Model ® Integration (CMMI SM), Version 1.1*, report CMU/SEI-2002-TR-011, Software Engineering Institute (SEI), 2002.

[5] Cummins F. A., *Enterprise Integration: An Architecture for Enterprise Application and Systems Integration*, ISBN 0471400106, John Wiley & Sons, 2002.

[6] Karolak D. W., *Global Software Development - Managing Virtual Teams and Environments*, ISBN 0-8186-8701-0, IEEE Computer Society, 1998.

[7] Land R. and Crnkovic I., "Existing Approaches to Software Integration – and a Challenge for the Future", In *Proceedings of Software Engineering Research and Practice in Sweden (SERPS)*, Linköping University, 2004.

[8] Land R., Crnkovic I., Larsson S., and Blankers L., "Architectural Reuse in Software Systems In-house Integration and Merge – Experiences from Industry", In *Proceedings of First International Conference on the Quality of Software Architectures (QoSA)*, Springer, 2005.

[9] Land R., Larsson S., and Crnkovic I., *Interviews on Software Integration*, report MRTC report ISSN 1404-3041 ISRN MDH-MRTC-177/2005-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, 2005.

[10] Meyers C. and Oberndorf P., *Managing Software Acquisition: Open Systems and COTS Products*, ISBN 0201704544, Addison-Wesley, 2001.

[11] Ruh W. A., Maginnis F. X., and Brown W. J., *Enterprise Application Integration*, A Wiley Tech Brief, ISBN 0471376418, John Wiley & Sons, 2000.

[12] Wallin C., *A Process Approach for Senior Management Involvement in Software Product Development*, Licentiate Thesis, Department of Computer Science and Engineering, Mälardalen University, 2003.

[13] Wallnau K. C., Hissam S. A., and Seacord R. C., *Building Systems from Commercial Components*, ISBN 0-201-70064-6, Addison-Wesley, 2001.

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