Software In-House Integration – Quantified Experiences from Industry

Rikard Land[1], Peter Thilenius[2], Stig Larsson[1], Ivica Crnković[1]

Mälardalen University, [1]Department of Computer Science and Electronics, [2]School of Business
PO Box 883, SE-721 23 Västerås, Sweden, Tel +46 21 10 70 35

{rikard.land, peter.thilenius, 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 consolidate the existing in-house developed software. There are many high-level strategic decisions to be made, which should be based on as good foundation as possible, while these decisions must be made rapidly. Also, one must employ feasible processes and practices in order to get the two previously separate organizations to work towards a common goal. In order to study this topic, we previously performed an explorative and qualitative multiple case study, where we identified a number of suggested practices as well as other concerns to take into account. This paper presents a follow-up study, which aims at validating and quantifying these previous findings. This study includes a questionnaire distributed to in-house integration projects, aiming at validation of earlier findings. We compare the data to our previous conclusions, present observations on retirement of the existing systems and on the technical similarities of the existing systems. We also present some practices considered important but often neglected.

1. Introduction

When organizations merge, or collaborate very closely, they often bring a legacy of in-house developed software systems, systems that address similar problems within the same business. As these systems address similar problems in the same domain, there is usually some overlap in functionality and purpose. It makes little economic sense to evolve and maintain these systems separately (this is true for any kind of system built internally, independent of whether they are core products offered to the market or are internally built tools mainly used in-house). A single coherent system would be ideal. This situation may also occur as systems are independently developed by different parts of the same organization; as they grow a point will be reached where there is too much overlap, and should be integrated. This paper presents the results of a questionnaire survey designed to study this topic, which we have labelled "in-house integration".

The questionnaire is based on earlier observations from an explorative qualitative multiple case study [29]. This previous study consisted of nine cases from six organizations. The main data source was interviews, but in several cases, we also had access to certain documentation. The previous material was analyzed from several points of view [16]:

- The influence of existing systems' architecture on what can be reused into a future system.
- Beneficial process practices and risk mitigation tactics, based on the interviewees' own descriptions of mistakes and successes.
- Available high-level strategies and the most important factors to exclude some of them.

The present paper presents a study in which a questionnaire was designed and distributed in order to validate and quantify the earlier observations.

Section 2 describes the research method employed. The questionnaire and the results are presented in Section 3. Section 4 relates this work to existing research, and Section 4 concludes the paper.

2. Research Method

Based on the results of the previously mentioned qualitative and explorative study, we constructed a questionnaire which was administered to people that had participated in in-house integration efforts. The questionnaire was constructed with some one hundred statements concerning their project, divided into four groups connected to our earlier findings: high-level strategies and decision-making, reuse and retirement, technologies and architectures of the existing systems, and process practices. All statements were to be graded with the same scale, ranging from 1 ("I do not agree at all") to 5 ("I agree completely").

2.1 Purposes

In relation to the previous multiple case study, there are three purposes of the questionnaire:

- **Validation.** By returning to the previous cases, some amount of internal validation of our previous interpretations (in terms of theory construction) is ensured. If the same cases are described in a very different way from our previous interpretations based on interviews, it is a sign that the theory is a bad representation of the reality. Also, by administering the questionnaire to some cases that were not part of the previous study, we get an indication whether the theory extracted from the previous cases makes sense at all.
- **Quantification.** Given that internal validity is satisfactory, some of the previous qualitative observations can be quantified. For example, the previous study led to a list of suggested beneficial practices, which can now be ranked in importance.
- **Hypothesis testing.** Our previous analyses includes some suggested correlations and causal links which can now be tested. This means that we search for uniform patterns across the respondents if the answers to two (or more) questions vary together.

2.2 Sampling Method and Response Rate

From a statistical point of view, the ideal situation would be to define the population (all in-house integration projects) and then sample randomly from it. However, although there exist databases with e.g. all companies in Sweden, there is no known way of find all newly merged companies. It would also be problematic and time-consuming to find the right persons within the selected companies, and there is no inherent reason to exclude non-commercial software developing organizations, or foreign organizations. The most resource-efficient method was to use convenience sampling, i.e. we identified potential projects and respondents through personal contacts.

Twelve cases were contacted, with a total of around 25 people, to ensure at least one response from most cases. We received responses from eight cases, nine people. The response rate was thus 2/3 of the cases, and ca 1/3 of the potential respondents. Our conclusions per case are therefore sensitive to individual responses, but conclusions where all responses are summed are less sensitive. We expect to continue distributing the questionnaire to more cases and respondents in the future, and the current data should only be seen as preliminary indications. The questionnaire and the full nine responses are published as a technical report [17]. The cases come from various domains, including safety-critical software, calculation-intensive simulation programs, and data-intensive systems. We make no analyses here concerning possible differences across domains or sizes of programs or organizations but refer to [17] for anyone who wants to draw their own conclusions. The naming of the cases is chosen as to be consistent with that of previous publications [16].

3. Results

This section is organized per question studied. Each subsection describes 1) the previous observations, 2) a summary of the questions in the questionnaire aimed at validating or quantifying these previous observations, 3) the most important questionnaire results, and 4) some interpretations of these results.

3.1 Integration Strategies

Based on the qualitative data from the earlier multiple case study, four high-level integration strategies have previously been described:

- **Start from Scratch** Discontinue all existing systems and initiate the implementation of a new system. The new system will likely inherit requirements and architecture from the existing systems.
- **Choose One** Evaluate the existing systems, choose the one that is most satisfactory, evolve it if necessary, and discontinue all others.
- **Merge** Take parts from the existing systems and integrate them to form a new system that has the strengths of both.
- **No Integration** Do nothing (mentioned to be complete).

The questionnaire contains a number of questions intended to identify the selected strategies in the cases. The purpose is twofold: first, the questionnaire results would indicate how well cases in reality fit within the proposed classification. Second, for the cases of the previous multiple case study, the questionnaire answers would reveal how well our earlier interpretations (in terms of selected strategy) match these more directed questions, i.e. the questionnaire can validate our previous interpretations.

3.1.1 Validation

Profiles that describe a set of responses that would match a specific have been prepared. For example, the statement “All existing systems is (or will be) retired” should for the *Choose One* strategy yield the response 1 (“I do not agree at all”), while for *Start from Scratch* the response should be 5 (“I agree completely”). Each case was then matched with the profiles. This was done by calculating the absolute difference between

the actual response and the strategy profile for each question, and calculating the sum for all questions.

Table 1: Comparison of earlier and current interpretations.

Case	Earlier Interpretation	Questionnaire Interpretation
A	*Start from Scratch*	*Start from Scratch*
B	*Choose One*	*Start from Scratch*
C	Changed decision from *Merge* to *Choose One* (which was implemented, with some reuse)	Impossible to single out one clear strategy
E1	*Start from Scratch*	*Start from Scratch*
E2	*Choose One*, but with some reuse (i.e. resembling *Merge*)	Difficulty to distinguish between *Choose One* and *Merge*
F2	*Merge*	Difficulty to distinguish between *Choose One* and *Merge*
F4	(This respondents' response concerned "the company's codes in general", so these responses do not qualify for identifying a chosen strategy)	
G	*Choose One* (based on a phone call; not previously published)	Impossible to single out one clear strategy

Five of the previous nine cases were revisited, and the previous interpretations and the questionnaire results were compared. The results are shown in Table 1. For cases A and E1 the results are in harmony with each other. For case C, E2, and F2 the questionnaire results do not clearly point at one single strategy, but not conflicting with previous interpretations. (Actually, our earlier interpretation of case C was that the vision was unclear and direction changed several times, which the questionnaire data might reflect; the respondent also made several notes in the margin of the questionnaire like "depends on what point in time". For case F2, which we interpreted as a *Merge*, there are actually several separate sub-systems that were earlier interpreted as *Choose One* and *Start from Scratch*; it is unclear exactly which system or subsystem the respondent had in mind when answering the questionnaire.) For case B, which we earlier considered a clear case of *Choose One*, the result is now a clear *Start from Scratch*. Returning to this particular respondent's questionnaire data, some of the questionnaire responses are exactly contrary to what was said during earlier interviews; we find no other interpretation than that some of the questions must have been ambiguous and misinterpreted (we believe the qualitative data closer to the truth).

Our conclusions must be: the questionnaire data are primarily in line with our earlier interpretations. However, many cases in reality would be described as hybrids (e.g. *Choose One* with some reuse). The strategies might still be useful as a model in discussions and planning. Also, the contradictory data given in cases B and G calls for refinement of the questionnaire before it is used in further research.

3.1.2 Decision Making Considerations

A number of questions were asked concerning how the high-level decision was made. The responses to each question are summed, and scaled to the percentage of the possible maximum (i.e. if all responses were "5"). The result is shown as a ranked list of statements in Table 2. The statements that scored highest are the ones with the most homogeneous answers, while for others opinions tend to differ more among the respondents.

Table 2: The relative importance of decision making considerations.

The high-level decision about how to integrate…	Importance (Percent)
...was made by management	84
…was based on technical considerations	71
…was based on considerations concerning the parallel maintenance and evolution of existing systems	63
...was made by technicians	62
…was based on considerations for existing users	60
…was based on available staff and skills	60
...was based on politics	60
...was based on considerations on time schedule	38

We see that management makes the decision rather than technicians, although they are also involved. The most important bases for decision seem to be of technical nature and concerns about the organization. Considerations on time schedule seem to be the least important consideration in in-house integration, which might indicate that such strategic tasks must be allowed to take the time and resources it requires; there is perhaps no perceived alternative. (Here we should note that any possible *No Integration* cases were probably systematically excluded from participation in the survey, as we initially searched for cases where integration had actually happened, or was underway.) The distribution of answers was generally larger for statements with lower importance.

3.2 Retirement of Existing Systems

We previously found the discussions concerning the impact of retiring the existing systems to be an important influence when selecting an integration strategy, and the questionnaire was designed to elaborate this somewhat. The exactly same statements used when comparing with the strategy profiles were to be graded both according to what was management's vision for the integration, and what was the actual outcome (or seem to be, if it is not finished yet). The largest differences between vision and

eventual result concerned retirement of systems, but interestingly the differences were in both directions: sometimes management wished to retire but this never happened, and sometimes management did not plan to retire but eventually some system(s) were retired nevertheless. The lesson learned is that one needs to pay extra attention to the issue of retiring systems.

The respondents were asked whose opinions the decision on retireability was based on, among customers, users, developers, marketing people, and management. Only small differences were found (with fairly large distribution), so the conclusion is that all of these seem to have been involved to the same extent.

There were also a set of questions concerning backward compatibility of the future system, i.e. whether the future system: 1) needs to support the way users currently work, 2) be backwards compatible with existing data, 3) be backwards compatible with existing surrounding tools, and 4) be backwards compatible with installations of the existing systems. No obvious difference was found; if anything, the last aspect seems to be the least important. The differences were large between the cases though: some cases scored high for all these four aspects, others low. This means that other factors such as market situation probably have a significant influence on these requirements, and that we based on our investigations are unable to formulate general guidelines.

Table 3: Similarities among the systems in the cases.

Statement	Similarity (Percent)
S: The existing systems contain software parts/components/modules with similar functionality.	78
TF: Communication between components/modules/parts in the existing systems is performed through certain interfaces.	66
S: The hardware topology (networks, nodes) of the systems is similar.	60
DM: The design of the existing systems is based on the data model.	60
TF: The existing software use the same or similar technologies.	56
DM: The data models in the existing systems are similar.	54
S: The existing systems interacts with the users in the same way.	53
TF: The existing systems use some technology to clearly encapsulate software components/modules/parts.	51
TF: The existing systems are written in the same programming language.	51
DM: The implementations of data handling in the existing systems are similar.	50
S: The existing systems have similar look-and-feel of the user interface.	49
S: The parts/components/modules exchange data in the same ways in the existing systems.	36
S: The software of the existing systems have the same internal structure (architecture).	35

3.3 Architectural Compatibility

We previously found the architectures of the existing systems to be important when selecting an integration strategy, and as described in section 3.1.2 technical considerations seem to be influential when deciding on a strategy. The respondents were asked to grade how similar the existing systems were according to a number of criteria.

The differences were large between the cases, and when summed across the cases (and scaled to the percentage of the possible maximum), clear differences can be discerned as to how common some particular similarities were in general. See Table 3. The questionnaire was designed so that each such criterion was considered part of either structure (S), data model (DM), or technology/framework (TF). When comparing these groups, no general differences can be identified.

The single similarity that stood out as the most common (with small distribution of answers) was that the existing systems contain software parts/components/modules with similar functionality. This should not be too surprising, since the systems in each case are in the same domain, solving similar problems. Both systems to be integrated should contain a user interface component, a physics calculation component etc. if they are both built to address a problem that requires a user interface and physics calculations. This can be compared with the statement "the software of the existing systems have the same internal structure (architecture)", which scored lowest of all criteria. This would mean that although there are components with similar roles in both systems (e.g. user interface, physics calculations, etc.), these are organized in different ways. Also scoring very low is "the parts/components/modules exchange data in the same ways in the existing systems". Together, this suggests that picking the best components from different systems and reassembling them into a new system (the Merge strategy) is most likely very difficult. This is in line with the well-known observation on "architectural mismatch" [9].

We also analyzed the connection between the three groups of similarities and three possible sources of similarities that we have previously identified. This is done by plotting the similarities and the responses in each case concerning possible sources of similarities (exemplified by Figure 1). Based on the current data, if the systems are initially built in the same time period, the technology/ framework and data models there seem to be a tendency of similarity (see Figure 1). If the systems implement the same domain standards, such as for safety-critical applications, this seems to be linked to all types of similarities. One previously reported

cause of similarities is when systems have been branched from the same system as a result of earlier collaborations. However, based on the questionnaire data it is impossible to claim such a link, which is somewhat surprising. The respondents might have interpreted "common ancestry" in the questionnaire differently than was our intention, and this needs to be considered in further research.

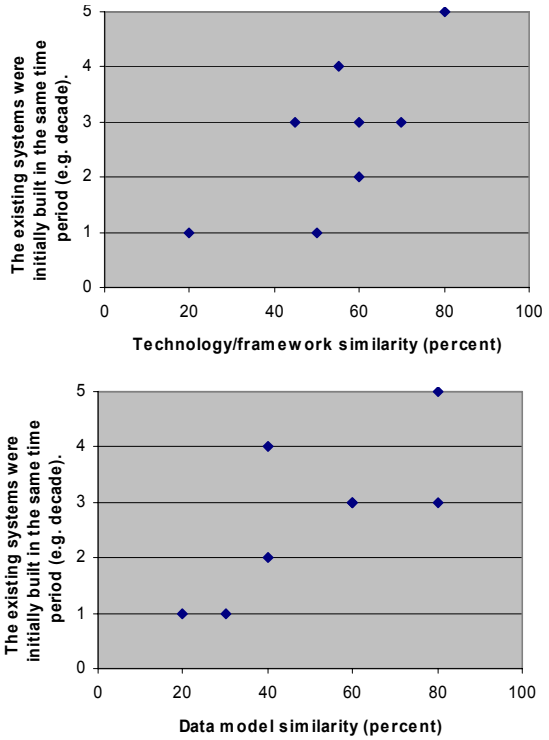

Figure 1: Indications of a link between same time of origin and similarities.

3.4 Exclusion of Strategies?

In earlier research we have presented the hypothesis that a certain amount of compatibility or similarity between the existing systems is a prerequisite for *Merge*, and that to be able to choose the strategies *Start from Scratch* and *Choose One* it must be considered feasible to retire (some of) the existing systems. The questionnaire was designed to be able to correlate the strategy questions with the compatibility questions and the retirement questions, but the low number of respondents and the weak support for describing some cases with a single strategy makes it difficult to draw any definite conclusions. Table 4 presents the cases in decreasing order of similarity, and we see that the cases that mostly resemble a *Merge* (as described in section 3.1.1) are at the top.

Similar questions, but fewer, were asked in order to investigate how discussions and decisions on retireability are linked to the choice of strategy. No clear difference can be found between the cases on how statements on retirement have been graded. Retirement will arguably depend on the situation on the market, and we still believe that retirement and choice of strategy must be discussed together. This means that the question is complex and the measurement instrument must be refined; it is probably necessary to more clearly distinguish between the message to the market and the plans for the actual implementations. For example, a completely new implementation could be marketed under an old, popular name.

Table 4: The similarities in the cases.

Case	Strategy Selected	Similarity: Average (S/DM/TF) (Percent)
F2	*Merge /Choose One*	**76** (77/80/70)
E2	*Merge /Choose One*	**70** (50/80/80)
A	*Start from Scratch*	**64** (63/60/70)
C	*Merge /Choose One*	**57** (72/40/60)
B	*Choose One / Start from Scratch*	**56** (47/60/60)
E1	*Start from Scratch*	**53** (33/80/45)
G	*Choose One* (Inconclusive)	**46** (50/35/53)

3.5 Process Practices

Based on the previous multiple case study data, the questionnaire listed a number of process practices and risk mitigation tactics. Compared to previous publications, these statements were somewhat shortened (and in some cases divided into several statements) to make them more straightforward to rank. The respondents were asked to grade: 1) how important this was for project success, and 2) how much attention it was given in the project. Both these grades were summed across the cases (and scaled to the percentage of the possible maximum). In Table 5, the practices are enumerated in decreasing order of importance. The attention given is also listed, as well as the difference between importance and attention.

The highest score, with the smallest spread among answers concerns the need of management's commitment, shown by allocating enough resources. The responses also show that a strong project management is needed, and that the grassroots – i.e. the people who will actually do the hard and basic work – must be cooperative, both with management and each other. Another practice that scored high was that a small group of experts should be assigned early to evaluate the existing systems and describe alternative high-level strategies for the integration. A number of practices scored somewhere in between, and the two that scored lowest in total (although with greatest distribution of answers) were that the future system must contain more features than the existing systems (which one would expect in order to sustain commitment for a long and expensive project), and that the future system should be described in terms of

the existing systems (which we previously found to be an intuitive way of working rapidly with requirements). That they scored lowest does not mean they are unnecessary, only considered less important when compared with other practices.

Table 5: Process practices.

Statement	Importance (Percent)	Attention (Percent)	Difference (Percent)
Management needs to show its commitment by allocating enough resources	**98**	58	40
A strong project management is needed	**95**	68	28
The "grassroots" (i.e. the people who will actually do the hard and basic work) must be cooperative, both with management and each other	**93**	78	16
A small group of experts must be assigned early to evaluate the existing systems and describe alternative high-level strategies for the integration.	**93**	74	18
Experience of the existing systems from many points of view must be collected.	**91**	73	19
All stakeholders must be committed to the integration	**89**	64	25
A common development environment is needed	**82**	70	12
Decisions should wait until there is enough basis for making a decision	**80**	58	22
Formal agreements between sites must be made and honored (strictly obeyed)	**80**	45	35
There is a conflict between the integration efforts and other development efforts	**77**	60	17
It is more important that decisions are made in a timely manner, even if there is not enough basis for making a decision	**75**	60	15
The future system must contain more features than the existing systems	**58**	58	0
The future system should be described in terms of the existing systems.	**54**	50	4

Even more interesting than studying perceived importance on its own, is to compare it with how much attention the practices received in the projects, as an indication of how often these practices are neglected. The greatest difference between importance and attention was found among the practices graded as most important. Ranked by this difference, together with the statements on commitment (from both management and grassroots) and the need for a strong project management, we find the need for more formal agreements between sites than usual. The fact that these four practices have not gained enough attention illustrate the challenges involved in coordinating two previously separate organizations with their existing local priorities, as well as the well-known challenges of distributed software development [5,6,11].

4. Related Work

The insight that software evolves, and has to evolve, is not new [18,21]. Software has to be extended in different ways to keep up with evolving needs and expectations, and interoperability and integration is one type of extension. There is literature describing integration rather fundamentally in terms of a) interfaces and interoperability [27,28], b) architecture [1,9,12] and architectural patterns [4,23], and c) information/taxonomies/ data models [10]. These foundations are directly applicable to the context of in-house integration.

The concept of integration is fundamental in the field of Component-Based Software Engineering [25,26]. Also, the notion of standard interfaces and open systems [20], builds upon the idea of integration in the sense interoperability. However, these existing fields address somewhat different problems than in-house integration. Integration in these fields means that components or systems complement each other and are assembled into a larger system, while we consider systems with overlapping functionality. The challenge in in-house integration is therefore not to assemble components into one whole, but to take two (or more) whole systems and create one single whole, containing the best of the previous systems. Also, these fields typically assume that components (or systems) are acquired from third parties controlling their development, meaning that modifying them is not an option. We focus on systems completely controlled in-house, and this constraint consequently does not apply.

Enterprise Application Integration (EAI) is a large business as well as a research area [8,13]. EAI is a systematic approach to integrating systems acquired from third parties, typically with technologies such as middleware and message buses. This is suitable since EAI focuses on data-centered systems, with high requirements on e.g. throughput and scalability. The systems subject to in-house integration could be other types of systems, such as control systems with resource constraints and requirements on safety and real-time behavior, which means that the EAI solutions do not apply. While EAI is a way to interconnect one specific set of installations together, in-house integration concerns integrating systems that may be deployed and installed millions of times. The focus shifts towards efficiency per installation, both in terms of man-time and runtime.

There are established software development models, ranging from the traditional sequential waterfall model with different variants [19], iterative, incremental, and evolutionary models [3,19], the commercially marketed Rational Unified Process

(RUP) [15] and agile methodologies [2,24]. The focus is usually on new development, although there is also literature on component-based development processes [7,26]. It is already known that reusing software is difficult [14,22], something confirmed also in the case of in-house integration. There is a body of knowledge on global and distributed software processes, also focusing mostly on new development [5,11]. Many identified challenges seem to hold also for in-house integration, e.g. overcoming physical distance and cultural differences [6].

5. Summary and Conclusions

This paper presented a study of in-house integration projects that validates and quantifies earlier observations. The low number of respondents means that interpretations must be very carefully made. Relating the current study with earlier research, the method that has been validated, and thus serves as suggestion for future projects, can be described as follows.

The notion of high-level strategies might be useful as a model during discussions, in that they represent the extreme ends of the solution space. However, no case is in reality easily described with one single strategy. This reflects that reality is more complex than models of it. Strategy selection is usually made by management rather than technicians, but is based on technical considerations. This might indicate that there is a higher awareness among managers of the potential technical problems than we have reported earlier. Based on the data, the least important consideration when selecting a strategy is the cost and time associated with the integration. This might indicate the strategic importance of integration. The option of not integrating is often not considered a real option, so the integration must be allowed to take time. In this part of the study, there were a few instances of responses that are contrary to earlier interviews with the same respondents, which call for further investigation and improvement of the measurement instrument.

We have found some support for the hypothesis that *Merge* is only possible if there is a certain amount of similarities. Three types of similarities are presented that should be investigated in future projects: structure, data model, and technology/framework. Across our cases, these types of similarities were roughly equally common, but within a case the types of similarities found differed greatly. The greatest difference in general is found between two measures of structure similarities: although it is common that there are components with the same roles in systems addressing the same problem in the same domain, the structure of these components is usually different (contrary to our earlier observations). The data indicates a connection between the time period when a system was built and similarities; some similarities can also be expected if there are domain standards implemented by the systems.

To achieve the necessary combination of technical knowledge and timely decision of a high-level strategy, a small group of technical experts should be gathered early; this practice was reported as important across the cases. Such a group would evaluate the existing systems and outline and analyze possible (and impossible) future, integrated systems. Other practices reported as important concern commitment of stakeholders – in management's case this must be accompanied with providing adequate resources. One important lesson is that there is a large difference for these and more practices, between the perceived importance and to what extent these are actually employed in reality. In particular, the challenge of making two previously separate organizations strive towards the same goal is often underestimated; one solution is to have strong management, make agreements between sites more formally than either site is normally used to in in-house projects, and to have a mechanism for enforcing these.

For future software in-house integration projects, there are several lessons to be learned. First, there are a number of important practices that are often overlooked; future projects should therefore ensure that these practices are employed. Second, approaches with much reuse from several systems (expressed as a *Merge* in the extreme case) must have a certain degree of similarity to succeed. Third, even if many of these results make intuitive sense, it might be a reassurance for future decision-makers to know that others have successfully made similar decisions.

5.1 Future Work

Our intention is to provide a refined analysis based on a larger number of cases. That would also make it possible to distinguish possible differences between e.g. application domains and sizes of organizations. We would also like to take part in projects where some of the current observations and suggestions are systematically employed and evaluated.

5.2 Acknowledgements

We would like to thank all respondents and their organizations for sharing their experiences and allowing us to publish them. Thanks to Laurens Blankers at Eindhoven University of Technology for previous collaborations in this research field. Thanks to Cecilia Erixon at School of Business at Mälardalen University for help with designing the questionnaire.

6. References

[1] Bass L., Clements P., and Kazman R., *Software Architecture in Practice* (2nd ed.), ISBN 0-321-15495-9, Addison-Wesley, 2003.

[2] Beck K., *EXtreme Programming EXplained: Embrace Change*, ISBN 0201616416, Addison Wesley, 1999.

[3] Boehm B., *Spiral Development: Experience, Principles and Refinements*, CMU/SEI-2000-SR-008, Software Engineering Institute, Carnegie Mellon University, 2000.

[4] Buschmann F., Meunier R., Rohnert H., Sommerlad P., and Stal M., *Pattern-Oriented Software Architecture - A System of Patterns*, ISBN 0-471-95869-7, John Wiley & Sons, 1996.

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

[6] Carmel E. and Agarwal R., "Tactical Approaches for Alleviating Distance in Global Software Development", In *IEEE Software*, 18(2), 2001.

[7] Crnkovic I., "Component-based Software Engineering - New Challenges in Software Development", In *Software Focus, John Wiley & Sons*, 2001.

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

[9] Garlan D., Allen R., and Ockerbloom J., "Architectural Mismatch: Why Reuse is so Hard", In *IEEE Software*, volume 12, issue 6, pp. 17-26, 1995.

[10] Guarino N., *Formal Ontology in Information Systems*, ISBN 9051993994, IOS Press, 1998.

[11] Herbsleb J. D. and Moitra D., "Global Software Development", In *IEEE Software*, 18(2), 2001.

[12] Hofmeister C., Nord R., and Soni D., *Applied Software Architecture*, ISBN 0-201-32571-3, Addison-Wesley, 2000.

[13] Johnson P., *Enterprise Software System Integration - An Architectural Perspective*, Ph.D. Thesis, Industrial Information and Control Systems, Royal Institute of Technology, 2002.

[14] Karlsson E.-A., *Software Reuse : A Holistic Approach*, Wiley Series in Software Based Systems, ISBN 0 471 95819 0, John Wiley & Sons Ltd., 1995.

[15] Kruchten P., *The Rational Unified Process: An Introduction* (2nd ed.), ISBN 0-201-70710-1, Addison-Wesley, 2000.

[16] Land R. and Crnkovic I., "Software Systems In-House Integration: Architecture, Process Practices and Strategy Selection", In *Information & Software Technology*, volume Accepted for publication, 2006.

[17] Land R., Thilenius P., Larsson S., and Crnkovic I., *A Quantitative Survey on Software In-house Integration*, MRTC report, Mälardalen Real-Time Research Centre, Mälardalen University, 2006.

[18] Lehman M. M. and Ramil J. F., "Rules and Tools for Software Evolution Planning and Management", In *Annals of Software Engineering*, 11(1), 2001.

[19] McConnell S., *Rapid Development, Taming Wild Software Schedules*, ISBN 1-55615-900-5, Microsoft Press, 1996.

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

[21] Perry D. E., "Laws and principles of evolution", In *Proceedings of International Conference on Software Maintenance (ICSM)*, IEEE, 2002.

[22] Poulin J. S., *Measuring Software Reuse: Principles, Practices, and Economic Models*, ISBN 0-201-63413-9, Addison-Wesley, 1997.

[23] Schmidt D., Stal M., Rohnert H., and Buschmann F., *Pattern-Oriented Software Architecture - Patterns for Concurrent and Networked Objects*, Wiley Series in Software Design Patterns, ISBN 0-471-60695-2, John Wiley & Sons Ltd., 2000.

[24] Stapleton J., *DSDM - Dynamic Systems Development Method*, ISBN 0-201-17889-3, Pearson Education, 1997.

[25] Szyperski C., *Component Software - Beyond Object-Oriented Programming* (2nd ed.), ISBN 0-201-74572-0, Addison-Wesley, 2002.

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

[27] Wegner P., "Interoperability", In *ACM Computing Surveys*, volume 28, issue 1, 1996.

[28] Wileden J. C. and Kaplan A., "Software Interoperability: Principles and Practice", In *Proceedings of 21st International Conference on Software Engineering*, pp. 675-676, ACM, 1999.

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