Interviews on Software Integration

Rikard Land, Stig Larsson, Ivica Crnkovic

Mälardalen University, Department of Computer Science and Electronics
PO Box 883, SE-721 23 Västerås, Sweden
+46 21 10 70 35

{rikard.land, stig.larsson, ivica.crnkovic}@mdh.se, http://www.idt.mdh.se/{~rld, ~icc}

Abstract

Integration or replacement of existing software systems is inevitable after company mergers and other types of collaborations between organizations. Our present research aims at describing how organizations have accomplished new generations of their existing systems by integrating, merging, or reusing parts of existing systems, be it code, design ideas, or requirements. As an important part of this research, a number of interviews with people from industrial integration projects have been carried out, as well as participation in one case. The present report describes the methodology used to set up these interviews and contains the full notes made during the interviews. No analysis of the results is made.

Keywords
Software Integration.

1. Introduction

It is commonly observed that software systems must be evolved in order to remain satisfactory and meet new requirements [19,23]. One particular kind of evolution that becomes more and more important is migration and integration of existing software (legacy) systems [3,20]. The present research focuses on the situation when an organization wishes to create a new generation of their software products based on several existing systems. 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. There are many decisions to be made, and many factors influencing the decisions, as indicated by the question marks in the figure.

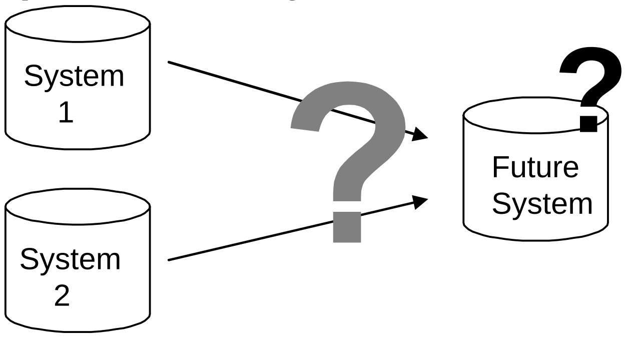


Figure 1: The challenges of integrating two systems into one.

The present research intends to study these decisions and factors *when the organization has total ownership and control* of the systems to integrate: source code, documentation etc. is available for modification without restrictions. Integrating systems developed by others, as e.g. is done during COTS (Commercial-Off-The-Shelf) component integration [4,10,27,30], or in the EAI field [5,13,20,21,26] has already been extensively studied, but we have identified the context presented here as basically non-existing in literature [17] while being a common and important problem in industry (indicated, if by nothing else, by the relative ease to find cases to study).

A multiple case study has been conducted, with the intent to study how this problem has been addressed in industry, and in what contexts it appears, as a part of our overall research in how integration can be successfully accomplished. After a case study revealing both the relevance and complexity of the problem [14], and a literature survey showing that the topic has not been previously researched [17], the purpose of the current phase is to get a broader overview of how the problem has occurred and been addressed in industry. The multiple case study involves 9 cases in 6 organizations.

1.1 Acknowledgements

Many thanks to all the interviewees and their organizations, which we are generally not allowed to mention by name. Also thanks to Jakob Axelsson for reviewing and helping improving the interview questions before the interviews took place.

1.2 Disposition

In section 2, the methodology of the research is presented in depth; the unit of analysis, data collection methods and threats. Section 3 introduces the organizations and the actual cases, and also lists the data sources for each case. Appendix 1 lists the interview questions used in the research, and Appendix 2 collects all interview notes. There are no analysis, summary or conclusions sections, as the purpose of the present report is to make available the raw data (the interview responses) and describe the research methodology in detail. Analyses of the material is planned to follow in separate publications.

2. Methodology

The research strategy chosen is that of a multiple case study [24,31], although it also bears resemblance of "grounded theory" research [24,29] due to the exploratory nature of the research. In grounded theory research, the researcher sets out with the goal of creating new theory, by investigating new cases (or the same case) to collect more data only as long as the data analyzed so far brings new insights. A multiple case study research design would aim at investigating enough cases (as many as possible) to give arguments convincing enough for a certain proposition. There is no strict border between these two strategies; in fact a multiple-case study appears to be a suitable approach for grounded theory research, only that the initial proposition is very vague, and the number of cases and the data needed for each case is not fixed at the beginning. In the following, the research can be regarded as "grounded theory research, but the focus is on the chosen strategy of a multiple-case study.

The goal with "real world research" is "to come up with a final set of research questions, which are relevant to the purposes of the study (which may, or may not, have been renegotiated along the way), which show clear linkage to theory (from whatever source it has been obtained) and for which the sampling has been such that the data you have collected and analyzed provide answers to those questions" [24]. Such a flexible design thus allows a certain amount of reformulation of research questions. In the case of the present research phase, the purpose is to gather experience from industry, the theory is not yet developed (although some observations were made during the previous participatory case study, and literature has been and will be further consulted), and the selection of cases are such that they are interesting, in the sense concerning large systems in large organizations.

2.1 Unit of Analysis

In this research, a case is not an organization, a system, nor a project, although these at first might seem appropriate characterizations. The best definition of a case is perhaps the resolution of an identified functional overlap between software systems within an organization. This means that several cases may be found within the same organization; in our research we have studied two or more cases within organizations E and F. The same people being interviewed may have experience from several cases; interviews I_{F1a} and I_{F2a} are made with the same person. The same system may also be the part of more than one case, which is true for case E1 and E2. The units of analysis are the different aspects studied contributing to an overall explanation and understanding of the cases: process and organization; requirements, functionality and goals of the integration; technical choices. With the terminology of Yin [31], this is a multiple case study, with multiple units of analysis.

2.2 Data Collection

Data collection was prepared by writing a set of interview questions, including a description of the purpose of the research (included as Appendix 1). This was in most cases distributed to the interviewees in advance of the interviews, although in most cases the interviewee had not studied it in advance. One of the authors (R.L.) prepared the questions, and one of the others (I.C.) and another senior researcher reviewed these questions before the interviews started. Interviews has been considered the main data collection method, common to all cases, but as described earlier, other sources of data has been used as well when offered.

To collect the data, people willing to participate in the interviews were found through personal contacts. The interviews were to be held with a person in the organization who:

1. Had been in the organization and participated in the integration project long enough to know the history first-hand.
2. Had some sort of leading position, with first-hand insight into on what grounds decisions were made.
3. Is a technician, and had knowledge about the technical solutions considered and chosen.

All interviewees fulfilled either criteria 1 and 2 (project leaders with less insight into technology), or 1 and 3 (technical experts with less insight into the decisions made). In all cases, people and documentation complemented each other so that all three criteria are satisfactory fulfilled. Interviews were booked and carried out. Robson gives some useful advices on how to carry out interviews in order to e.g. not asking leading questions [24], which we have tried to follow. In some cases, the interviewees offered documents of different kinds, which are listed in section 3 below, “Introducing the Projects”.

2.3 Addressing Threats

There are many threats to a sound research design. These are often categorized into construct validity, internal validity, external validity, and reliability ([31], p.34). The threats are discussed below, together with an account for the measures taken to address them.

2.3.1 Construct Validity

Construct validity concerns ensuring that the objects being measured and the measures used really highlight the concepts being studied. Yin gives three advices ([31], p.34), which are followed as described below:

- **Use multiple sources of evidence.** For some of the cases, there are two or more interviews, and in some cases there are additional information as well (documentation, and/or personal experience with the systems and/or the organization). For others, one interview is the only source of information (which is clearly a deficiency). To some extent, this can be excused by the exploratory nature of the research, and also that we wanted to find a proper balance between the number of cases and the depth of each.
- **Establish chain of evidence.** The exploratory nature of the research makes this advice somewhat less applicable, at least until data analysis and reporting. During these phases however, it will be made sure that conclusions are traceable to data, and that all data is considered in the conclusions.
- **Have key informants review draft case study report.** This has been done, in the form of a workshop where some of the interviewees discussed pending results.

2.3.2 Internal Validity

With internal validity is meant making sure any data contributing to the conclusions are properly assessed and documented, so that e.g. not spurious relationships are mistaken for true causes and effects. Three threats to internal validity are [22,24]:

- **Description.** It is essential that the descriptions of data (e.g. interview notes) are accurate. In case of the present research, this is addressed through "member checking", i.e. letting the interviewees review the copied out interview notes [24].
- **Interpretation.** Being open-minded and allow the data to explain themselves without imposing a framework upon them is essential, although it can be argued that this is impossible in practice – a researcher's previous experience will certainly be an important element during the interpretation. As the present research aims at building theory, this threat is somewhat less applicable.
- **Theory.** Alternative theories that may explain the same events should be considered. This is also somewhat less applicable to the current, theory-building research phase.

Triangulation is an important means in general to achieve internal validity [24,31]. Data triangulation means that multiple sources of evidence are used; in the case of the present research this means interviewing several people in the same case, using other types of sources in some cases (documentation and personal experience). Also, letting interviewees check the interview notes is a valuable means to avoid researcher bias and increase internal validity [24,31]. Triangulation may also take the form of observer triangulation, methodological triangulation, and theory triangulation [6], of which observer triangulation has been used in one case so far (case C_D), where a fellow researcher participated during the interview, and the copied out notes were reviewed by him.

2.3.3 External Validity

External validity concerns establishing the domain to which a study's findings can be generalized. Yin's main advice ([31], p.34) for multiple-case studies is to use "replication logic". The interview questions used in all interviews are the same, and during each interview it has been ensured that all are responded to a non-trivial extent, but no more than seemed appropriate for the case. Moreover, the underlying issues were in the mind of the researcher during each interview, and the interesting issues were pursued.

To generalize the results, the concept of theoretical replication [31], will be used. This means that even though the data of the different cases are not the same (which would be labeled literal replication [31]), they might point at the same theory. For example, if in one case the interviewee says "we did *X* and it was good because…", and in another case a similar response is "we did not do *X*, but in retrospect we think we should have"[1], this points at the same theoretical proposition, that doing *X* is beneficial. Analytical generalization [24] means the same thing, but the opposite term would be statistical generalization (which we have argued is not possible with a multiple-case study).

2.3.4 Reliability

Reliability concerns the repeatability of the study, and involves establishing and reporting many practical data collection issues. Any one studying the same case should be able to repeat the data collection procedure and arrive at the same results (at least unto some point of subjective interpretation). Yin's two advices ([31], p.34) and their application to the present research is described below:

- **Use case study protocol.** The case study protocol can be described as a workflow:
  1. Keeping track of who refers to who until an appropriate person in an appropriate project to interview is found.
  2. Booking and carrying out the interview. Interview notes are taken.

[1] Or "we did the opposite of *X* and think it was a mistake"; the opposite of doing *X* depends on what *X* is.

3. As soon as possible the notes are copied out (sometimes the same day, but in some cases more than two weeks afterwards).
4. The copied out notes are sent to the interviewees for review. This has several purposes: first, to correct anything that was misunderstood. Secondly, to consent to their publication as part of a technical report (considering confidentiality – in several cases the organization do not want to be recognized). Third, in several cases some issues worth elaborating were discovered during the copy-out process, and some direct questions are typically sent along with the actual notes. (In case C_B, the written response occupied more space than the complete interview response as noted by the researcher!) These thus reviewed, modified and approved notes are used as the basis for further analysis.
5. After some initial analysis, the interviewees (and a few other people from the same organizations, or who have otherwise showed interest in this research) have been invited to a workshop where the preliminary results are presented and discussed, giving an extra stage of feedback from the people with first-hand experience.

- **Develop case study database.** All notes (even scratch notes on papers, notes taken during telephone calls etc.) are kept in a binder for future reference. Also, any documentation is put in the same place. In order to be able to achieve the workflow described above, the stage of the workflow for each (potential) case is informally noted in an Excel sheet (date of last contact, action to be done by whom). All copied out interview notes are stored in a CVS system.

3. Introducing the Projects

This section introduces each interview. For each, the organizational history is briefed, as well as the basic technology and architectures in place. Also, the interviewee’s position in the company is presented.

In some cases, we were not allowed to reveal the name of the organization, nor any details that would reveal its identity. Descriptive names have therefore been assigned to the projects to hopefully make them easy to remember throughout the report. In Appendix 2, the complete notes from all interviews are collected. The cases are presented (here and in Appendix 2) in the order they were carried out.

To be able to refer to the organizations, cases and the resources (interviews, documents and participation) with short-hand abbreviations, the organizations, cases and resources have been named according to the following scheme:

- **The organizations** are named with the letters A to F (in approximately the order the interviews were carried out).
- **The cases** are assigned the same letter as the organization in which the case occurs. When there is more than one case in the same organization, this letter is followed by a number to make the case name unique: *X*1, *X*2 etc.
- **The interviews** are assigned the acronym I_X where *X* is the short name of the case. When there is more than one interview for the same case, the acronyms I_{Xa}, I_{Xb} etc. are used.
- **The documents** are assigned the acronym D_X where *X* is the short name of the case. When there is more than one interview for the same case, the acronyms D_{Xa}, D_{Xb} etc. are used.
- **Participation activities** are assigned the acronym P_X where *X* is the short name of the case. When there is more than one participation period for the same case, the acronyms P_{Xa}, P_{Xb} etc. are used.

3.1 Organization A

The company is an international company resulting from several company mergers. The physical products developed are large and complex, and have strong safety requirements. The development organization is distributed globally, and delivers generic solutions to particular sales projects, focused on particular products to particular customers. Each end delivery is very large, and is customer and project specific.

Case A: Developing Next-Generation HMI for a Real-Time Control System

The case concerns the current development of a “next generation” human-machine interface (HMI) for what could be characterized as an operator station. The control system uses the field bus to be predictable, and the operator station(s) communicate (mostly read-only) via a gateway connected to the field bus, to not interfere with the real-time behavior of the safety-critical parts. The communication to operator stations and other peripheral devices, such as a GSM transceiver, enabling remote analysis of the system, is based on Ethernet.

Resource:

- **Interview $\mathbf{I_A}$.** The interviewee is project leader for a “next generation” development project within the development organization.

3.2 Organization B

The company is the company within a large industrial concern engaged in developing, evolving, installing and maintaining business (information) systems.

Case B: Two Competing Information Systems within the Same Corporation

During the sixties, large information systems were used to rationalize administration. This case concerns the growth of one in-house administrative system for keeping track of hardware items kept in stock, and different types of integrations made during 40 years of operation. This case focuses on the case where another similar system, but developed for slightly different purposes, was to be integrated.

Resource:

- **Interview $\mathbf{I_B}$.** The interviewee is an experienced manager and developer, recently retired by age. He led the development and implementation of the system that eventually replaced the other in the case.

3.3 Organization C

In the late nineties a large international company, based in Sweden, acquired a US based company in the same business domain. The company develops control systems, including controllers at the low level, a set of mediating servers at the middle level, and operator stations. The organization has a large development organization (~600 people, including both hardware and software).

Case C: Merging Core Products: Distributed Embedded Real-Time Systems

This case concerns the events during recent years, when after the company acquisition there was a wish to market, maintain and support only one main product. Both companies had started developing a new generation of their systems at the time of the merger, and both was released, sold to customers and installed before a decision was made. The decision was to discontinue one of the systems and continue developing the other.

Resources:

- **Interview $\mathbf{I_{Ca}}$.** The interviewee is employed at the corporation’s research organization. He led the group consisting of five people investigating different integration alternatives.
- **Interview $\mathbf{I_{Cb}}$.** The interviewee is the main architect behind the Swedish system. He was part of the group investigating different integration alternatives.

3.4 Organization D

An international company, based in Sweden, acquired a US based company in the same business domain. The products control high-voltage electricity and are business- and safety-critical.

Case D: Merging Core Products: Reusing an HMI for a Server Application

After the company merger, there was a vision to in the long term merge the off-line analysis and management tools. This vision has not yet been concretized, and currently the relationship between the two software development units is strictly commercial (each tries to maximize its own profit). This has included reuse of the Swedish human-machine interface (HMI) in the US system.

Resources:

- **Interview I_{Da}.** The interviewee is architect and developer mainly of the server side of the application.
- **Interview I_{Db}.** The interviewee is architect and developer of the HMI.

In addition, one of the authors (S.L.) has informally taken part of some other people's version of the events. These sources cannot be published, but there is nothing that challenges the descriptions given by the interviewees.

3.5 Organization E

Cases E1 and E2 concern the same system, SILVIA, but during different periods. This system was developed and evolved as cooperation between the Swedish Defense Research Agency ("Försvarets Forskningsanstalt", FOA, later "Försvarets Forskningsinstitut", FOI) and industrial partners. The system is used to simulate air defence at different levels of detail.

Case E1: Cooperation Project for Next-Generation Physics Simulation

The events of case E1 occurred in the early eighties and concern the birth of the system. A number of previous simulation models for air defense at different levels had been developed by both industry and FOA, and there was a wish to integrate these. The existing models were implemented in different languages, but a new three-level architecture was defined, and the new generation would be implemented in Ada.

Resources:

- **Interview I_{E1}.** The interviewee was project leader and main interface developer of the project during the time period of the case.
- **Document D_{E1}.** The protocol of the startup meeting (1982) [28]. Provided by the interviewee.

Case E2: Possible Integration of Physics Simulations System

The events of case E2 occurred during the second half of the nineties, when SILVIA and two other systems were identified to have a certain amount of functional overlap. These were investigated and candidates for some sort of integration. The outcome was that the two other systems in practice were retired. The functionality was seldom used and was not transferred to SILVIA, the exception being the GUI of one of the other programs which is now used in SILVIA.

Resources:

- **Interview I_{E2}.** The interviewee was the project leader and main developer for SILVIA at that time.
- **Document D_{E2a}.** A comparison of the three simulating models in use, and a proposal of future development of the systems [1]. The conclusion was to continue development of the three systems in parallel, and in the long run develop a new generation of (a common) system. However, it is acknowledged that financing this is difficult. Provided by the interviewee. (In Swedish.)
- **Document D_{E2b}.** An overview of the air defense model SILVIA (1995) [9]. Provided by the interviewee. (In Swedish.)
- **Document D_{E2c}.** "The need of SILVIA from 2001", a document describing expectations on SILVIA and how it could, or should not, incorporate these new requirements [8]. Provided by the interviewee. (In Swedish.)
- **Document D_{E2d}.** "SILVIA – Description of the computer model" (2003), technical documentation [12]. Provided by the interviewee. (In Swedish.)
- **Document D_{E2e}.** "Experiences from using the assessment model SILVIA" (1997) [7]. The document presents experiences from SILVIA. The validity of the model is discussed, e.g. pointing out that the scenarios used for simulation have not always been representative for actual scenarios, therefore leading to results contradicting common sense. Provided by the interviewee. (In Swedish.)
- **Document D_{E2f}.** "Swedish experiences from Air defense models", a research paper describing the experiences with SILVIA [2]. Provided by the interviewee. (In English.)

3.6 Organization F

In late 2000 a large US enterprise acquired a smaller Swedish company in the same domain. Part of the business involves making advanced physics simulations to ensure safety. There are therefore high QA requirements on the software. Software development is not a large portion of the company's business, but should be seen as an important support activity. The main user base is internal users, but there are also external customers.

Case F1: Common Environment for Physics Simulations

Shortly after the merger, the two previously separate software departments tried to identify ways of rationalizing their activities and define a common strategy. The functionality of software for managing data related to physics simulations has been found to overlap to some extent. At the Swedish site, a data management system has been developed and used for some years, a system closely coupled to the 3D simulator. At the US site, several minor systems have been developed for similar tasks; two were in place at the time of the integration, and a third development project was run during 2004 with the goal of implementing a new data storage mechanism for the US 3D simulator. At several occasions, efforts have been made to outline a common vision of a future, common system, but so far no significant progress has been made.

A note on terminology: simulations are made for two main types of (real physical) environments, here encoded as types "A" and "B". Due to different customer needs, the A type has traditionally been the most common in Sweden and the other in the US.

Resources:

- **Participation P_{F1a}.** In late 2002 one of the authors (R.L.) participated as an active project member when several integration alternatives were developed and evaluated.
- **Participation P_{F1b}.** Fall 2004 to spring 2005 one of the authors (R.L.) participates as an active project member, developing a new storage mechanism for the future environment. This includes many things difficult to document: participation in project meetings, less formal discussions with other project members. Although research notes are currently being taken we have chosen to not publish them.
- **Interview I_{F1a}.** The interviewee is the main architect behind the Swedish data management system. This person is the same as in interview I_{F2a}.
- **Interview I_{F1b}.** The interviewee is the main architect behind two of the US systems. The interview mostly concerns one of them, a recent project aimed at providing a new data storage mechanism for the US needs, and its relation to the long-term integration.
- **Interview I_{F1c}.** The interviewee is QA responsible for the Swedish software department and project leader for the Swedish data management system.
- **Document D_{F1a}.** The experiences from participation during 2002 (P_{F1a}) were described from three different points of view in three conference papers [15,16,18]. Part of this material was based on a questionnaire responded to by some of the project members. Some additional analysis was made in the author's Licentiate thesis where these papers were reprinted [14].
- **Document D_{F1b}.** As a participating member, all project documentation is available.

Case F2: Next-Generation 3D Physics Simulations

For the actual simulations, each site has one main simulator (here called the 3D simulator). For both systems, there have been a chain of programs preparing a certain kind of input to (one part of) the simulator. These are described by the interviewees as a 2D simulator, before which a preprocessor is run, aiding in setting up the complex input data required by the 2D simulator, and after which a post-processor (often referred to as "linker" by the interviewees) is being run, taking the output from the 2D simulator and preparing it for use by the 3D simulator. Both the 2D simulator and the post-processor involve advanced physics and mathematics.

At the time of the company merger, both the US site and the Swedish site had a chain of programs implementing the sequence described; only the Swedish side previously did not have a pre-processor. Not long after the merger, this was identified to be subject to integration and rationalization. After more efforts than expected, the pre-processor, post-processor and module in the 3D simulators are now common. The future vision is to phase out the Swedish 2D simulator and merge the 3D simulators.

Resources:

- **Interview I_{F2a}.** The interviewee is a software engineer at the Swedish site and contributed, together with interviewee I_{F2d}, to the post-processor. This person is the same as in interview I_{F1a}.
- **Interview I_{F2b}.** The interviewee is a software engineer at the Swedish site and contributed, together with some US people to the pre-processor.
- **Interview I_{F2c}.** The interviewee is project manager for Swedish site (the overall project leader is located at the US site).
- **Interview I_{F2d}.** The interviewee is a physics expert at the Swedish site and contributed, together with interviewee I_{F2a}, to the post-processor.
- **Interview I_{F2e}.** The interviewee is a physics expert at the Swedish site and is the main author behind the new model implemented as part of the 3D simulator.
- **Interview I_{F2f}.** The interviewee is a software engineer responsible for the evolution of the Swedish 3D simulator.

Case F3: Implementing a Common Issue Tracking System

Three different software systems for tracking software issues (errors, requests for new functionality etc.) were used at three different sites within the company. There was an effort to integrate these tools into one, and use the best approaches with each. Eventually, a commercial system was acquired, to which all data was transferred.

Resources:

- **Interview I_{F3}.** Project leader and main implementer for the project.
- **Document D_{F3a}.** M.Sc. thesis authored by the interviewee, describing the initial analyses and efforts which later resulted in the issue tracking systems integration [25]. Provided by the interviewee.
- **Document D_{F3b}.** Some PowerPoint presentations the interviewee had held during the project. Some of these were consulted during the interview, and some figures of the presentation are included in the interview.

4. References

[1] Ahlqvist H., *Simulering av luftvärnsstrid*, Enskild uppsats LHU 19100:4054, Försvarshögskolan, 1997.

[2] Berglund E. and Hansson M. B., “Swedish Experiences from Air Defense Models”, In *Proceedings of American Institute of Aeronautics and Astronautics, Inc.*, American Institute of Aeronautics and Astronautics, Inc., 1999.

[3] Brodie M. L. and Stonebraker M., *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*, Morgan Kaufmann Series in Data Management Systems, ISBN 1558603301, Morgan Kaufmann, 1995.

[4] Crnkovic I. and Larsson M., *Building Reliable Component-Based Software Systems*, ISBN 1-58053-327-2, Artech House, 2002.

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

[6] Denzin N. K., *The Research Act: A Theoretical Introduction to Sociological Methods*, Prentice-Hall, 1989.

[7] Hansson M. B., *Erfarenheter från användingen av värderingsmodellen SILVIA (Experiences from using the assessment model SILVIA)*, report FOA-R--97-00559-202--SE, Defence Research Establishment, Div. of Guidance & Control, Materials and Underwater Sensors, 1997.

[8] Hansson M. B., *Behovsinventering av SILVIA 2001 (The need of SILVIA from 2001)*, report FOI-R--0117--SE, FOI - Swedish Defence Research Agency, Systems Technology, 2001.

[9] Harling S., Kallstenius S., and Westrin P., *Luftvärnsmodellen SILVIA - En översikt*, Försvarets Materielverk,Robotavdelningen, 1995.

[10] Heineman G. T. and Councill W. T., *Component-based Software Engineering, Putting the Pieces Together*, ISBN 0-201-70485-4, Addison-Wesley, 2001.

[11] IEEE, *IEEE Guide to Classification for Software Anomalies - Description*, report IEEE Std 1044.1-1995, IEEE, 1995.

[12] Johansson J. O., *SILVIA - Datormodellbeskrivning (SILVIA - Description of the computer model)*, report FOI-R--0858--SE, FOI - Swedish Defence Research Agency, Systems Technology, 2003.

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

[14] Land R., *An Architectural Approach to Software Evolution and Integration*, Licentiate Thesis, Department of Computer Science and Engineering, Mälardalen University, 2003.

[15] Land R., “Applying the IEEE 1471-2000 Recommended Practice to a Software Integration Project”, In *Proceedings of International Conference on Software Engineering Research and Practice (SERP'03)*, CSREA Press, 2003.

[16] Land R. and Crnkovic I., “Software Systems Integration and Architectural Analysis - A Case Study”, In *Proceedings of International Conference on Software Maintenance (ICSM)*, IEEE, 2003.

[17] 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.

[18] Land R., Crnkovic I., and Wallin C., “Integration of Software Systems - Process Challenges”, In *Proceedings of Euromicro Conference*, 2003.

[19] Lehman M. M. and Ramil J. F., “Rules and Tools for Software Evolution Planning and Management”, In *Annals of Software Engineering*, volume 11, issue 1, pp. 15-44, 2001.

[20] Linthicum D. S., *Enterprise Application Integration*, Addison-Wesley Information Technology Series, ISBN 0201615835, Addison-Wesley, 1999.

[21] Linthicum D. S., *B2B Application Integration: e-Business-Enable Your Enterprise*, ISBN 0201709368, Addison-Wesley, 2003.

[22] Maxwell Joseph A., “Understanding and validity in qualitative research”, In *Harvard Educational Review*, volume 62, issue 3, pp. 279-300, 1992.

[23] Parnas D. L., “Software Aging”, In *Proceedings of The 16th International Conference on Software Engineering*, pp. 279-287, IEEE Press, 1994.

[24] Robson C., *Real World Research* (2nd edition), ISBN 0-631-21305-8, Blackwell Publishers, 2002.

[25] Rosén G. and Samson F., *Design of an Information Portal for Engineers*, M.Sc. Thesis, Department of Computer and Information Science, Linköpings universitet, LITH-IDA-EX--04/040--SE, 2004.

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

[27] Sametinger J., *Software Engineering with Reusable Components*, ISBN ISBN: 3-540-62695-6, Springer, 1997.

[28] Sjunnesson G., *Mötesanteckningar LVRBS III Modellverksamhet*, report SBMC/Gus - 1/82, SAAB Bofors Missile Corporation, 1982.

[29] Strauss A. and Corbin J. M., *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (2nd edition), ISBN 0803959400, Sage Publications, 1998.

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

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

Appendix 1: Software Integration Discussion Material (used during interviews)

The present research is intended to investigate *integration of software systems*. Of interest are integration of systems that have a significant complexity, both in terms of functionality, size and internal structure, and which have been released and used in practice. Our research question is: what are feasible *processes* (such as when and how were different people involved in the process) and *technical solutions* (for example, when is reuse possible, and when is rewrite needed) to accomplish a successful integration?

The working hypothesis is that both processes and technical solutions will differ depending on many factors: the fundamental reasons to integrate, as well as the domain of the software, the organizational context, and if there are certain very strict requirements (as for safety-critical software). We aim to identify such factors and their importance. In particular, we are interested in the role of *software architecture* (that is, the systems' overall structure) during integration.

The questions below will be used rather informally during a discussion/interview, and are to be used as a guide. Preferably, the respondent has considered the questions in advance. It is not necessary that all terms be understood. There may also be other highly relevant topics to discuss.

1. Describe the technical history of the systems that were integrated: e.g. age, number of versions, size (lines of code or other measure), how was functionality extended, what technology changes were made? What problems were experienced as the system grew?

2. Describe the organizational history of the systems. E.g. were they developed by the same organization, by different departments within the same organization, by different companies? Did ownership change?

3. What were the main reasons to integrate? E.g. to increase functionality, to gain business advantages, to decrease maintenance costs? What made you realize that integration was desirable/ needed?

4. At the time of integration, to what extent was source code the systems available, for use, for modifications, etc.? Who owned the source code? What parts were e.g. developed in-house, developed by contractor, open source, commercial software (complete systems or smaller components)?

5. Which were the stakeholders[*] of the previous systems and of the new system? What were their main interests of the systems? Please describe any conflicts.

6. Describe the decision process leading to the choice of how integration? Was it done systematically? Were alternatives evaluated or was there an obvious way of doing it? Who made the decision? Which underlying information for making the decision was made (for example, were some analysis of several possible alternatives made)? Which factors were the most important for the decision (organizational, market, expected time of integration, expected cost of integration, development process, systems structures (architectures), development tools, etc.)?

7. Describe the technical solutions of the integration. For example, were binaries or source code wrapped? How much source code was modified? Were interfaces (internal and/or external) modified? Were any patterns or infrastructures (proprietary, new or inherited, or commercial) used? What was the size of the resulting system?

[*] Stakeholders = people with different roles and interests in the system, e.g. customers, users, developers, architects, testers, maintainers, line managers, project managers, sales persons, etc.

8. Why were these technical solutions (previous question) chosen? Examples could be to decrease complexity, decrease source code size, to enable certain new functionality.

9. Did the integration proceed as expected? If it was it more complicated than expected, how did it affect the project/product? For example, was the project late or cost more than anticipated, or was the product of less quality than expected? What were the reasons? Were there difficulties in understanding the existing or the resulting system, problems with techniques, problems in communication with people, organizational issues, different interests, etc.?

10. Did the resulting integrated system fulfill the expectations? Or was it better than expected, or did not meet the expectations? Describe the extent to which the technical solutions contributed to this. Also describe how the process and people involved contributed – were the right people involved at the right time, etc.?

11. What is the most important factor for a successful integration according your experiences? What is the most common pitfall?

12. Have you changed the way you work as a result of the integration efforts? For example, by consciously defining a product family (product line), or some components that are reused in many products?

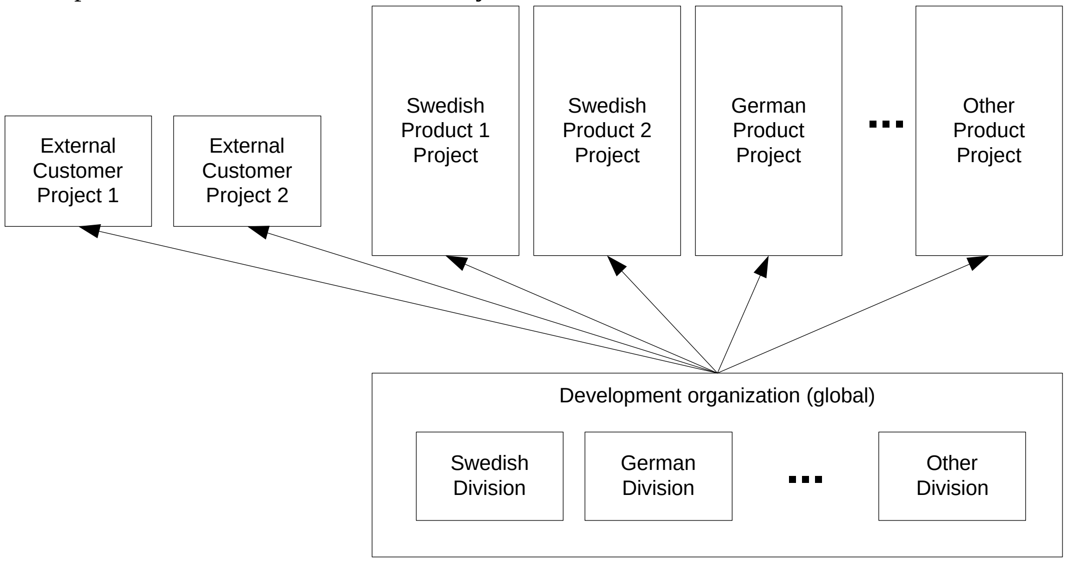
Appendix 2: Interview Notes

In this appendix, the complete notes from each interview are reported. In each case, the notes were edited by the present author to consist of complete sentences etc., and in some cases to remove details. The resulting text was then reviewed by the interviewee, and with minor modifications from them, the result is reported here.

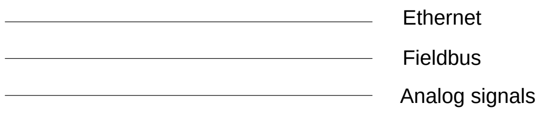
Interview I_A

1.

The company is an international company resulting from several company mergers. The development organization is considered global, and delivers generic solutions to particular sales projects, focused on particular products to particular customers. Each end delivery is very large, and is customer and project specific to some extent, but build on relatively standardized products. The development organization consists of several development divisions, which were previously the separate companies' development organizations. In some cases, the development also sells solutions and subsystems to external customers.


External Customer Project 1
External Customer Project 2
Swedish Product 1 Project
Swedish Product 2 Project
German Product Project
...
Other Product Project
Development organization (global)
Swedish Division
German Division
...
Other Division

The products developed are large, complex, and heavy, with strong safety requirements. There are mainly three types of communication solutions within these products, namely an analog signal system, a field bus (a digital data bus with predictable real-time behavior), and the most recent trend, an Ethernet bus based on IP. There have been different focus and traditions in how these types of communication solutions are utilized within the different parts of the company, partly stemming from different cultures and countries within the company, and different standards of different countries.



The interview concerns the development of a "next generation" human-machine interface (HMI) for what could be characterized as an operator station. The control system uses the field bus to be predictable, and the operator station(s) communicate (mostly read-only) via a gateway connected to the field bus, to not interfere with the real-time behavior of the safety-critical parts. The communication to operator stations and other peripheral devices, such as a GSM transceiver, enabling remote analysis of the system, is based on Ethernet.

There were a number of existing HMIs, developed and evolved by previously different companies. Different focus on the three different types of data communication (analog signal system, real-time field bus, and Ethernet). Different hardware and software platforms. One of these, developed in Sweden. Built with a commercial operating system.

Within a German delivery project, a new German platform around year 2000, none of the existing HMIs were considered sufficient. New HMI built on an open source OS (a strong movement in Germany).

New development project, in second half of 2003, by the company's development organization worldwide. Goal: new HMI that would replace the old ones. This development has been led by the Swedish part of the development organization, which was considered historically strong in the HMI part of the overall system. Currently (Sept. 2004) a prototype has been delivered (satisfactory) and the first production version will be finished in the middle of 2005.

2.

Previous HMIs (and complete systems) developed by different companies which later merged (somewhat simplified).

German development made within a delivery project (mostly delivery money, only little development money). The customer was German, and the German part of the company ran the delivery project. Strong contact with customer, concrete requirements, led to the development of a new HMI instead of using existing HMIs already developed and used in other delivery projects. It seems likely that the German development team wanted to show the abilities of their own ideas, not least the predisposition of open source, to drive development within the company globally along this path. The result was promoted as a candidate for a new generation of HMIs, to be used consistently throughout the company worldwide.

After this, the development organization worldwide decided to indeed create a new generation of HMI, to be used in all subsequent delivery projects worldwide. This development (funded mostly by development money) has been led (and mostly carried out) by the Swedish part of the development organization, which was considered historically strong in the HMI part of the overall system.

3.

One main reason was to have one single HMI to be used in many deliveries, thus lowering the costs for each delivery.

In order to achieve this, the solution had to be highly configurable so as to be able to cope with the variability of each delivery project. Compare with product lines, or framework. Two main types of configurability:

- The visual user interface (which meters etc. to display on screen) had to be configurable to meet the requirements of actual delivery projects with customers of many kinds, and different kinds of products, from many countries.
- The gathering of data from different sources must be possible to configure for different types of signals and data, as well as the type of diagnostics database. The hardware would be similar, though.

At the same time, the solution chosen would enable new features, like connecting a laptop or PDA to the HMI system for e.g. diagnostics and maintenance, which was seen as a valuable extra possibility.

4.

Total ownership by company globally.

5.

Stakeholders were sales people and management, presumably knowledgeable about end customers requirements (end customers meaning people buying the final products). Also engineers, used to configure and install HMI in the delivery projects. Mostly Swedish people were asked for their requirements,

It was explicitly asked for 10-20 "killing arguments", not a complete list of requirements, as it was expected to be too hard to synthesize too detailed requirements from all people involved. It was argued that an HMI developed based on these "killing arguments" would fulfill the product organizations' requirements to 90%, which would make them "enough" satisfied. It was furthermore argued that having too many too detailed requirements from all stakeholders involved would make it likely that the project would never come to an end, not even the requirements elicitation phase. The most important of these “killing arguments” were (without prioritization):

- Low cost. Total cost of ownership was discussed, but in reality the cost associated with a delivery project was the most important.
- Stability, meaning that the HMI should very rarely malfunction or crash, as it is part of a safety-critical system. (Interviewers note: “dependability” is probably the term used in research literature nowadays depicting this system property.)
- Long lifetime and/or simple maintenance. The complete product is expected to be in use for a long time, and so should the HMI. However, some computerized hardware, such as screens, has a relatively short life compared to many other types of hardware traditionally used. It must therefore be very easy to exchange parts simply by plugging out the old part and plugging in the new.
- Configurability was important, influenced by the graphical programming model IEC-1131, but this could be seen rather as a design decision than a requirement in itself, a means to meet the other requirements.

6.

The development team asked other technology companies in other domains with similar hardware architecture and high-level requirements, because it seemed that other domains had more cost-efficient solutions. Such domains included industrial robots and large forest machines. Both the hardware topology and technology was investigated, as well as the requirements in those domains on their data presentation (HMI).

The Swedish development of the next generation chose a development model where a prototype was shown to management to gain approval for the complete development plan. Today, the prototype has been successfully demonstrated and the project proceeds. The requirements and design of several previous HMIs were known through the people participating in the project.

The German HMI had shown the Swedish development team the importance and value of a configurable HMI. Interviews took place both within the Swedish part of the company on requirements (see also under question 5), as well as with the German developers of the German HMI.

Although asking more people throughout the organization (globally) would give more information and more requirements, this was not done. Several similar projects had previously stranded because there were too many requirements from too different sources to synthesize. One can ask where the optimal tradeoff is between succeeding in finishing such a project within time and budget, and the number of stakeholders to be involved to ensure the result being acceptable throughout the organization worldwide. It was moreover considered that the development team (and the Germans interviewed) and the Swedish product organizations were representative, or had enough knowledge of requirements worldwide.

Many requirements were stated in terms of existing HMIs: “Should look like X”. Also, the data available on the field bus was already given (although different from product to product).

7.

Mostly the HMI involves new software. Interfaces: hardware, overall architecture.

The software is based on commercial technologies from a major vendor: a commercial operating system with a tight connection to a component model, with the associated (commercial) development tools. A component-based approach has been chosen, based on the component technology. This enables modularization of the code so that e.g. different communication “back-ends” can be developed and used without modification of other parts of the HMI.

Reuse is mostly in the form of the features of the existing HMIs (the basic functionality that have to be there, the idea of configurability) and some design decisions (e.g. configurability) – and also some anti-design decisions (i.e. learning from what was not so good with previous HMIs).

There was no structured effort to reuse code, but intuitively this seemed the wrong way to save efforts, since the requirements were partly new, and a new platform (operating system, component model, development tools, etc.) was chosen. However, ideas were taken from existing HMIs: many requirements were at a high level the same as previous HMIs, and design choices as well, not least the idea of configurability.

8.

The commercial path was chosen to a large extent due to the history of the Swedish development organization. Previous Swedish HMIs had been based on Visual Basic, so it was natural to continue with .NET; also, it was very important to be able to run under Mono, a .NET implementation for Linux. This was a way to merge two different development paths.

Familiar technology is not only a matter of what people like, but also gives a better foundation for cost and schedule prediction, and a lower risk compared to using technologies with which the organization is unfamiliar.

A component-based approach with support from a component technology brings certain benefits, such as a modularized and easy-to-understand system.

9.

So far, the prototype has been delivered as expected. The first functional product will probably be later (half a year) than initially expected.

The initial task was to come up with a demonstrator to prove the approach, after that the intention was to reschedule the project according to the result of the demonstrator.

10.

To demonstrate the feasibility of the approach from a customer point of view, existing HMIs have been modeled utilizing the high configurability. The existing user interfaces of other HMIs have been modeled in great detail, with colors, sizes of controls, behavior, etc. This prototype demonstration was considered satisfactory by management, and continuation of the development was approved.

There are no major concerns about the project plan.

11.

Most important factors:

- Skilled team members.
- Clear requirements, and not too many with too much details. 10-20 “killing arguments”, together with the experience of previous HMIs in the organization, and knowledge of customers, is sufficient.
- The development team should be able to work without disturbance according to the project vision. There will inevitably be turbulence around the project – why not do this way, or that, why this technology, why not these requirements, why the Swedish development organization and not the German, etc. The project leader should drive the development towards the initial goal, and show results externally, instead of modifying goals or involving the team members too much in the debate on whether the project is feasible or not.
- Move project members close to each other to ease daily contact.

One lesson learned is that the requirements should have been even clearer to begin with.

12.

The domain is such that each delivery project is very customer-centered, so configurability is a key issue to become cost-efficient. Previous HMIs were not very configurable, that is, not before the German HMI. Also, the component-based approach is more pronounced now than in previous HMIs.

Interview I_B

1 & 2.

In the second half of the sixties, administrative systems for keeping track of hardware items of all kind kept in stock were developed within the large Swedish industrial corporation ASEA. At this time, there was much talk about “totally integrated systems”, influenced by American universities. This approach was tried at ASEA too, but abandoned in 1967 after half a year – higher management had the feeling that this was not a sound investment. It was perceived that it would take too long time to finalize such a system. Indeed, some other Swedish companies tried this approach, unsuccessfully – at another large company, the building of such a totally integrated system lasted for 15 years, without ever being finished. The major fault with this approach is that it is impossible to take a snapshot of the business and the organization and go implement an administrative system for 5 years or more – businesses change too rapidly. It was also difficult to know the business and the organization as the processes were only in the minds of people. Also, there is a too long time to return on investments.

Instead, the approach used was to create systems one at a time, where the largest rationalizations (on the user sides) could be made by using computer programs. In this way, the number of human administrators could be reduced from (at least in one case) approx. 100 persons at a department to less than one. There was no clear strategy for future integration, the goal was to rationalize. There were some 40 computer program developers who made things in similar ways and standardized “in the corridor”. Thus, things were made in similar but not identical ways in different implementations. The older manual systems were well known and used as templates for how things should be made.

At first, the system used punch cards, was run in batches once a week, and the items in stock were presented on long lists to the users. The manual system used cards for each item and was immediately updated. The computer system – presenting old data in a format difficult to read – was therefore not accepted until terminals with online access were introduced in the second major version (five years after first version).

As several such administrative systems were used throughout the corporation, and a need to integrate these was identified. Integration (described in more detail below) occurred through a number of years until the beginning of the eighties, when the integration was fairly all-encompassing.

VIPS-Y was a major, large, administrative system, which had been built according to the “totally integrated system” paradigm in parallel with AROS (the loosely integrated system, built piecewise in order to rationalize).

Most recently, there is a decision that the system will be replaced by commercial ERP systems, a growing trend during the nineties. This change in direction has so far been extremely costly for the corporation. This is due to several reasons. Partly, more licenses than needed have been acquired. … This represents a step back to the totally integrated system paradigm, which is not a viable approach.

3.

The main reason to integrate was to rationalize further, by decreasing administrative staff and sharing information between different systems.

For VIPS-Y: as time passed, there were new requirements on functionality that was supported by AROS. VIPS-Y was piecewise replaced. Each such decision was not very dramatic, and concerned only a few people in the business. The guarantee that the information in VIPS-Y was transferred to AROS without interrupting the business was decisive.

4.

All systems were completely developed and owned by the corporation.

5.

The user organizations (or the corporation at large) had an interest in having a system that made rationalizations possible.

The users wanted to have at least the same functionality (up-to-date information accessible in a convenient format) as the previous, manual systems.

In the developing organization, there was a long term interest to do things similarly when developing different systems, in order to make integration in the future smooth. This was done mostly informally, in the corridor.

6.

The choice of integration was somehow pre-determined, since things were made in similar ways in order to enable integration in the future. Inter-connecting this kind of administrative systems by adapting to some provided interfaces of a host system is somehow the way it should be done.

7.

One system was chosen to be the main database system for stock keeping. All other systems had to adapt to this system's interface (the interface was at this time in the form of punch cards). Adapters were often written to accomplish this, i.e. to take the output from one system, calculate and enter missing data.

Standardization was made when need demanded it. In this way, development could be relatively rapid, without need for thinking about future integration. Of course, there were problems stemming from this when integration actually took place. For example, there could be the same concepts and terms in different systems, but defined differently.

AROS was from the start written in assembler and PL/1, and is still running on mainframe computers. There has never been a motivation to port them.

There was also the case of VIPS-Y, the totally integrated system, that should now be integrated into AROS, the loosely integrated, overall system. There was a certain overlap in many functions. VIPS-Y had not been modularized in order to interact with other solutions. That was not the mission of that project – the knowledge was not there. This integration took a number of years, and VIPS-Y was stripped down piece by piece, and functionality rebuilt within the framework of the loosely integrated system. The same people were involved, and the existing structure had degraded since many people were involved, each putting their own "touch" to the system. Many design ideas were reused from the totally integrated system.

8.

To enable information exchange. Maintenance rather became more complex since additional wrappers were written (more code and more complicated overall structure).

9.

Each integration of one system into the overall integrated system (where the main database system for stock keeping was the hub) was carefully planned, and the actual integration took place during a weekend. These integrations always went as planned (only on one occasion was the system unavailable to the users during Monday before noon).

VIPS-Y, the totally integrated system was replaced piecewise. In this way, experience was gathered on how to carry out integration. Each change was made during weekends, without stopping production.

10.

The integrated system today has a history of about 40 years, and not a single time has a system failure stopped production. It has been extremely reliable throughout all these years. It is very heterogeneous (“many islands”) and consists of 10-15 core systems, and 100-150 smaller systems. The system is very well documented.

However, it is difficult to maintain due to the many systems and the few people involved – previously there were 3-4 people per system, while now there are 6-7 for the whole system. It is also considered not modern (mainframe and PL/1). The commercial ERP systems are by corporation management considered a more viable approach. The drawbacks with that solution is that you have no control over the evolution of such a product, and that you still have to build a lot of custom solutions on top of the system.

11.

Most important: to understand what the customer wants. The benefit in this case was that the developers worked closely with the customers and users (within the same corporation, originating from the very same organization before computerization). Also, planning and care is essential.

12.

The many similar integrations of systems into the large integrated system became more and more routine. That is, until the choice of using commercial ERP systems. Either you believe in a homogeneous totally integrated system, or not.

13. The interviewee’s own notes, after the feedback loop (in Swedish).

“Take a snapshot” kan tolkas på två sätt:

Det **första** är kanske det som är tydligt ovan – att speca hur ett system skall stödja verksamheten och sedan förverkliga det. Framkallningstiden är för lång, många hinner komma in i mörkrummet och störa tillverkningen. Verkligheten utvecklas snabbt och i oförutsatta banor. Systemutvecklarnas förmåga, iver och flexibilitet räcker inte till. När dörren till mörkrummet öppnas så blir främlingskapet uppenbart. Kraven på systemen har hunnit ändras – nya processer, produkter och människor.

Det **andra** är att det heller inte fanns någon bra kamera (metod och verktyg) att ta ett snapshot med. Verksamheten var sällan dokumenterad. Processer och rutiner fanns i huvudet på människor. Bilden som systemutvecklarna tog blev suddig och ofullständig. Man bortsåg också från att processhastigheten i ”verksta’n” var mycket större än i datorerna. ”On-line” fanns inte med i ordförrådet.

Man insåg inte att det varken fanns möjlighet att ta ett adekvat snapshot eller att framkalla det på rimlig tid. Systemutvecklarnas optimism var då, precis som nu, förödande för ett sunt samspel mellan tekniken och behoven.

En bidragande orsak till att avbryta tankarna på ”totalintegrerat” måste ha varit några mycket kostsamma och misslyckade statliga systemutvecklingsprojekt. Det fanns på ASEA folk som förstod att ”datorfolket” varken hade koll på sitt eget område eller på verkligheten. Ungefär 35 år senare hade ”datorfolket” vuxit i styrka och vann några slag, men …. (”IT-bubblan” som sprack).

Under åren 1965-67 gjordes även en verklig utveckling av ett ”totalintegrerat” för Y-sektorn (elektroniksektorn, embryot till en av nuvarande ABB’s hörnpelare). Huvudsyfte var att skapa bättre och snabbare rutiner inom sektorns verkstäder och förråd. Projektet gick under beteckningen Y66. För att få snabb rapportering av händelser i verksta’n utvecklades bl.a. ett antal rutiner med hålkortsterminaler (några av världens första terminaltransaktioner!). Den första versionen togs i drift 1967. Systemet döptes då till VIPS-Y (Verkstadsintegrerat Produktions-System för sektor Y). Tanken var att systemet skulle utvecklas vidare för andra sektorer. Systemversionerna VIPS-O, VIPS-S etc. skulle följa i rask takt.

VIPS-Y hade god effekt på Y-sektorn. Kostnaden för rutinstödet till verksta'n antog dock helt nya dimensioner och stor tveksamhet om lönsamheten var allmänt utbredd. Under Nicolins VD-skap fick Y-ledning och dataavdelning motivera lönsamheten för systemet minst en gång om året under 60- och 70-talet. Faktum är att man alltid lyckades med detta. De sista delarna av VIPS-Y togs ur drift först i slutet av 80-talet.

Det gjordes ett försök att utveckla ett VIPS-O, men detta kom inte igenom nålsögat för genomförande. Minnet sviker mig hur det negativa beslutet motiverades. Troligen hade det flera orsaker. VIPS-Y hade utvecklats helt med tanke på behoven inom Y. Systemkonstruktionen hade inte gjorts moduliserad för att senare kunna kombineras med andra lösningar. Förklarligt eftersom det inte var projektets uppdrag – kunskapen fanns inte. Behoven inom O-sektorn var annorlunda och nya systemdelar måste utvecklas för att nå önskad rationalisering. Vi får heller inte glömma att konceptet för verkstäder då höll på att ändras radikalt mot ”produktverkstäder”. Majoriteten av de för produkten nödvändiga resurserna gjordes tillgängliga och styrbara av produktverkstaden. Stora ansträngningar för att reducera ställtider gjordes. Specialverkstädernas långa genomloppstider (frosseri i köer) blåstes undan.

ASEA genomgick under dessa år en utveckling från stora centrala produktionsenheter till sektoriserade enheter. Sektoriseringen var till en början helt produktionsinriktad. Styrningen av ASEA förändrades, centraliseringen bröts upp och ett antal nya produktionsenheter skapades med olika behov av styrning och processer. Förändringen ledde till en ny syn på hur datautvecklingen bäst skulle bidra till företagets lönsamhet.

Inom ramen för projektet Y66 utvecklades ett system för specifikation av produktionsorder (dokumentation av struktur, operationer, förrådsmaterial, inköp, verktyg etc.). Systemet hade beteckning SPEC-66. Till grund för detta system låg en mycket väl utvecklad standard för ritningssystem, där kraven på mångfald i strukturbehov (konstruktion, produktion, montage, underhåll, reservdelar etc.) är tillgodosedda. Ett centralt grepp om konstruktionssystematik och standardisering upprätthölls inpå 90-talet.

SPEC66 blev senare SPEC-ADB (när AROS-ROSAM togs i bruk i början på 70-talet) och ett av de mest lönsamma systemen i AROS-familjen. Det är från detta systems implementations-framfart, som exemplet med rationaliseringsresultat 100 till < 1 för en implementering, är hämtat. SPEC-ADB kunde införas sektorsvis eller verkstadsvis oberoende av andra system och skapa rationella lösningar. SPEC-ADB hämtade grunden från VIPS-Y och hade en mycket viktig komponent i databasnycklarna – ”användarkod”. Koden gav möjlighet till individuell anpassning av systemets funktioner. Det var möjligt att köra en gemensam systemuppsättning för många användare. Oerhört viktigt för att hålla drifts- och underhållskostnader på låg nivå.

Användarkod i databasnycklar blev regel i all systemutveckling. En av de absolut viktigaste förutsättningarna för ett långt systemliv i en mycket dynamisk företagsmiljö (40 års drift !).

1967 påbörjades utvecklingen av systemstöd inom materialsidan (förråd, lager, inköp). Projektet gick under namnet CM-systemet (Centralt Materialstyrsystem). Systemen utvecklades från specifikationer gjorda av Centrala Materialavdelningen (ansvarig för inköp- och förrådsverksamheten). Första versionerna av förrådssystemen togs i drift 1969. Stegen var stapplande de första åren. Långsamma och dyra datorkörningar (batch en gång i veckan) kunde inte slå kartotekskortens omedelbara aktualitet. Övertron på operationanalysens praktiska möjligheter var stor (behovsprognosering och beräkning av optimala orderstorlekar var dåligt utvecklad för den typ av produktion som är förhärskande inom ASEA – orderbundet i små, sällan återkommande orderkvantiteter) och förhoppningar att nå en ökad precision materialförsörjningen med mindre kapitalbindning drev på.

När andra versionen togs i drift med möjligheter till online-transaktioner (AROS-ROSAM 1972) och ökad frekvens i batch-körningarna började systemet ge effekt. Förrådssystemet CM-F/L implementerades successivt på samma sätt som SPEC-ADB.

Fördelarna med en systemvis implementering var uppenbar. Vi kunde begränsa systemutveckling till det mest önskvärda och få snabba återbetalningstider. Alla investering motiverades med snabbt (återbet ca 1 år) uppnåbara rationaliseringar (reduktion av mantid).

Vänder vi för ett ögonblick tillbaka till SPEC-ADB så gav det två uppenbara effekter. Grunddataregistret CASO (Centralt Artikel Struktur och Operationsregister) gjorde det mycket enklare att hålla hög datakvalitet. Det omfattande handarbetet med att ta fram orderspecifikationen ersattes av maskinellt framställda listor och hålkort. Även om hålkortsstansningen krävde en insats så förenklades specarbetet och registerunderhållet dramatiskt. Inga integrationer till andra system gjordes vid spectillfället. Återrapportering till redovisningssytem blev enklare eftersom korten var förstansade med identifierande uppgifter.

Fortfarande fanns alla underlag för tillverkning på verkstadskontoren. Ett omfattande arbete krävdes för att underhålla underlaget när något ändrades i ordern.

När CM-F/L infördes så uppnåddes liksom för SPEC-ADB fördelar med registerunderhåll. Problemet med långsamma batch-system blev här kritiskt. Kraven på integration med SPEC-ADB kom snabbt. Detta och högre körfrekvens ökade installationsfarten. Integrationen gjordes som hålkortstransaktioner. CASO blev basregister för att underhålla artikeldata och SPEC-ADB levererade orders behov av förrådsuttag. Datorkörningarnas komplexitet ökade och nätterna räckte ibland inte till. Mycket stor andel av handarbetet med att hålla kartoteks och bokföringsuppgifter aktuella hade eliminerats.

CM-INKÖP för att stödja administration av köp från externa leverantörer startade försiktigt 1969. Liksom för systemen ovan var första steget att förenkla framställningen av inköpsdokument. Förhoppningen var också att kunna bevaka moment i händelsekedjan från beställning till faktura. Hålkortstekniken med batchkörningar räckte inte till för detta. Det blev heller inte enklare av att kraven på decentralisering av inköpsverksamheten började komma. Möjligheterna till uppdatering och frågor via terminaler blev nyckeln till succé. 1972 gjordes de första ankomstrapporteringarna via bildskärmsterminal och utskrift av ankomsthandlingar utanför den centrala materialavdelningen. Utvecklingen av systemet gick nu snabbt. Systemet gav möjlighet att organisera processerna på ett helt nytt sätt. Det gick att välja mellan central och decentral administration av olika moment i inköpets händelsekedja. Systemet behövde inte ändras när organisationen ändrades. Integrationer med CASO, SPEC och CM-F/L och redovisningssystemen gav snabba resultat.

Gemensamt med alla integrationerna var att de byggde de i varje system existerande hålkortstransaktionerna. De enskilda systemen växte funktionellt och volymmässigt. Kraven på snabbhet i integrationerna gjorde att natten ibland inte räckte till.

I samma anda utvecklades ordersystem som kombinerade försäljningsarbetet med konstruktionssystemen och sedan förde data vidare till produktionssystemen. Produktverkstäderna satte fart på kraven och snart fanns kompletta maskinella informationsvägar från säljare till verkstadskontor.

Systemen utvecklades raskt mot komplett uppdatering via terminaltransaktioner. Detta gav en möjlighet att minska trycket på batchkörningar och få snabbare rutiner. I st f att skicka hålkort till varandra började system sända terminaltransaktioner. Samma princip som den ovan med hålkort, men mycket snabbare. Under AROS-systemens kulminering gjordes ca 1,5 millioner transaktioner i genomsnitt varje dygn.

Det var först nu som tekniken var tillräckligt bra för att klara ett stöd för produktionsarbetet. Ett system, PROMAC, som tog vid där SPEC-ADB slutade, utvecklades (1982). Informationen som fanns i hålkortshögarna på verkstadskontoren gjordes tillgänglig via terminaler. Dokumenten framställdes först när arbetet i verkstan skulle påbörjas. Aktualiteten i informationen fick ett kraftigt uppsving. Händelser rapporterades on-line. Informationen för både den dagliga planeringen och den långsiktiga styrningen blev goda underlag för snabbare genomlopp, minskade störningar och lägre kapitalbindning. Nu blev systemkedjan komplett från konstruktion, försäljning, inköp, produktion och redovisning. Möjligheterna att ersätta VIPS-Y fanns.

Detta arbete hade redan påbörjats. CM-systemen integrerades med VIPS-Y och ersatte delar av detta. När PROMAC togs idrift för ”Y-sektorn” så kunde ett lyckat systemprojekt pensioneras. VIPS-Y lade grunden till mycket av den senare systemutvecklingen. SPEC-ADB med sitt CASO hämtades till stora delar från pionjärarbetet med SPEC66. CM-F/L hämtade inspiration men nyutvecklades från grunden. Opertionsanalysens frukter hämtades från IBM-system. CM-INKÖP hade ingen motsvarighet i Y66 och det fanns heller inga inköpssytem på marknaden. Systemet blev en stark pådrivare för nyvecklingen av terminaltrafiken. Systemet för att hålla reda på orderstocken i verkstäderna, REBUS, hade mycket att idéerna att hämta från VIPS-Y.

När det var dags att ”skrota” VIPS-Y hade systemstödet för ”Y-sektorn” utvecklats inom många områden. Det fanns ett antal stora system för konstruktion och produktion som var kopplade till VIPS-Y. Förutom att bryta ut och ersätta funktioner i VIPS-Y med PROMAC gjordes ett antal nya integrationer till dessa system. Vi fick alltså en rejäl erfarenhet av successiv förädling av systemstödet för en verksamhet. Vi hade även ständigt kraven på oss att aldrig stoppa systemen. Alla förändringar skedde under helger för att vara igång vid sjutiden måndagmorgon.

”Fajten” mellan VIPS-Y och AROS pågick under många år. Det gällde för AROS att visa på fördelar både funktionellt och kostnadsmässigt. AROS hade fördelen av att ett antal renodlade system i portföljen. Vi kunde byta ut en del av VIPS-Y i sänder. Besluten blev därför ganska odramatiska och berörde ett fåtal personer i verksamheten. Garantin för att informationen i VIPS-Y fördes över i AROS smidigt och utan driftsstopp var avgörande argument. Skillnaden mot det senaste decenniets implementationer av totalintegrerade paket (från t ex SAP och Baan) är minst sagt påtaglig.

Ytterligare två faser i AROS integrationshistoria (under 90-talet) skall nämnas: Intåget av PC och kraven på snabba bokslut.

Den stegvisa uppbyggnaden av systemstöd som gjordes under ett antal decennier skapade ett stort antal system. Systemkedjan som skulle resultera i ett månads- eller årsbokslut blev mer och mer komplex. Detta krävde genomloppstider i datorerna och tid för avstämning mellan systemen. Vi hade minskat tiden från ca 15 dagar till 10, mest tack vare effektivare datorer. Standardsystemens säljare påstod sig klara mycket kortare tider och argumenterade för att snarast byta ut det omoderna AROS. Kraven på AROS ökade och genom ett nära samarbete med ekonomifolket kunde vi snart klara systemkedjan på 3 dagar. Återigen bar det nära samarbetet mellan kravställare och dataavdelningen frukt. Inga avbrott, ingen borttappad information, bara snabbare.

PCn kom tidigt att bli ett kraftigt slagträd mot stordatorsystemen. AROS snabba död spåddes.

Vi beslöt oss att försöka skapa en ersättare till AROS byggd på den nya och lovande tekniken.

Vi skulle göra en ny systemgeneration och byta ut AROS successivt på samma sätt som vi tidigare hade ersatt VIPS-Y. Vi kom en bit på vägen med PEGASUS, men nådde inte tillräckligt stor acceptans hos våra kunder. Principiella beslut hade fattats om att gå över till ett antal standardsystem. Efter 7 års drift blev PEGASUS ersatt av SAP. Även i PEGASUS gjorde vi integrationer som tog vara på AROS-systemen. Trafiken mellan AROS och PEGASUS var omfattande.

Efterklok kan man undra om vi inte borde satsat mer på att förändra användargränssnittet i AROS och utnyttja fördelarna med det grafiska gränssnittet i PC. Tekniken för att koppla ihop AROS-ROSAM med PC-tekniken utvecklades under senare delen av 90-talet. Ett antal lösningar där man från en PC kan kombinera flera av AROS-systemens information togs fram. Integrationen fick en ny dimension. AROS-systemens trygghet i drift och säkerhet kunde behållas och kombineras med ett grafiskt gränssnitt.

90-talet innebar stora förändringar. Decentraliseringen drevs framåt. Ett stort antal bolag bildades. De centrala staberna splittrades. Varje bolag fick egna datachefer. Bolagiseringen slog pendeln kraftigt mot att ”göra själv” och skaka av sig beroende av andra enheter inom koncernen. Naturligtvis gällde detta även systemstödet. Standarsystemen sågs som en lösning på oberoende och möjlighet till nya rationaliseringar och minskade kostnader.

Några bolag tog täten och installationer både Baan (Triton) och SAP (R/3) påbörjades. Systemen var ju inte framtagna med tanke på traditionerna inom ASEA-ABB. Funktioner som ansågs viktiga fanns inte. Nya begrepp, nya tankebanor frestade på i förändringen. Försök att påverka leverantörernas systemutveckling gjordes. Avtal skrevs. Stor frustrationen hos båda parter. ABBs ville snabbt bli hörsammat när det gällde krav på ny funktionalitet. Leverantörerna trodde att ABB var en guldgruva, många och lika installationer över världen. Så småningom (3-4 år) sansade sig alla och det gjordes ett antal hyfsat lyckade implementationer som fungerade (lång tid, stora kostnader både för implementation och drift, ingen lönsamhet – men väldigt strategiskt). Det krävdes omskolning och omtänkande i massor.

Den första riktigt lyckade installationen av SAP gjordes på ”Y-sektorn”. Bolaget var stort och komplext med många olika verksamhetstyper. En mogen syn på projektets komplextet fanns. De bästa krafterna mobliserades. Här gällde det inte bara att ”kasta ut AROS”. Man accepterade en stor del av kostnaderna redan under projektplaneringen. Man beredde sig på nya lösningar, omtänkande och uteblivna funktioner. Vi kan se likheten med när AROS ersatte VIPS-Y, men också olikheter. Systemkomplexiteten hade ökat. SAP kunde inte ersätta allt, utan det fanns en mängd system som skulle fortsätta att användas och kopplas ihop med SAP. AROS stegvisa klipp kunde inte upprepas. SAP måste tas i sin helhet vid ett tillfälle.

AROS ersattes. Vissa funktioner blev bättre, men många eleganta lokala lösningar försvann.

Kostnaderna fick ett nytt lyft (skalan måste ändras). Underhållet blev även mer komplext. Vid första omorganisationen kom en nytt ”paradigmskifte” – mer än ett halvt år att genomföra och åtskilliga miljoner fick byta plånbok.

AROS-systemen på ABB kommer att vara helt ersatta under 2006. Tyvärr har då ersättningen blivit ett antal sofistikerade ”totalsystem” från några 80-talisters drömfabriker i Ede och Walldorf. En mördande trögrörlighet har byggts in administrationen. Naturligtvis går det över. Kraven på flexibilitet kommer att finna nya lösningar.

Interview I_{Ca}

1 & 2.

A Swedish company and a North American company in the same business merged in the late nineties. Both systems were being the developed and not yet released.

The Swedish product was under development but not yet released. It was a new generation of the same type of core product the company had historically been developing and selling. The American product also was being developed and about to be released. The Swedish system introduced more novelties than the American. The American system was closer to a release than the Swedish system.

3.

The benefits were from the developing company's point of view in terms of cost: the rest of the development, as well as future maintenance costs would be decreased. Also, there was a business image concern – it seemed irrational to offer two similar products. The reason for buying the North American company was to gain access to the North America market

4.

The systems were completely developed and owned by the merged company.

5.

Customers are very conservative. Their primary concern is quality, in the sense stability and reliability. Graphical layout of operator stations should be clear and easy to read, and similar to previous generations of systems.

For the developing organization, the systems represent the core products. Higher management wants to provide innovative solutions and reduce costs to stay competitive, while of course reliability and safety are major concerns. Actual developers are attached to their systems, in the sense they like their own solutions mostly because they are their own. In the event of integration (due to rationalization), the developers also have an interest in keeping their own system, or at least their own development organization, in order to keep their jobs.

6.

The first decision was made in half a year after the company merger by higher management. This decision meant that the integrated system would consist of the underlying architecture (in the sense "platform") of the Swedish system and the user interface of the American system. It was perceived that the most advanced underlying functionality was presented by the Swedish system, and the user interface of the American was more mature. The integrated system should be delivered in less than a year after the decision.

After this decision, a technical group consisting of five people was appointed to elaborate what this solution would look like. The members were two Swedish technicians, two American managers knowledgeable in the technology, and a Swedish project leader (the interviewee). This constitution reflects a difference in business culture between America and Sweden. This group worked for a little less than half a year and elaborated different alternative ways of using the architecture of the Swedish system and the user interface of the American. The systems were not so modularized and easy to integrate. In their work, they used a computer with both products installed and used as a reference when designing different solutions. Much of the time spent was needed to understand the others' system.

A requirements specification had been made by the product management organizations, to be used both for the development of the American and the Swedish systems. This was finished at about the time when the technical group started their work. It was not considered useful in the process however, since it consisted of a very large number of requirements (about thousand) in a long list. They were prioritized, but there were still too many requirements with the highest priority to get an overview. It did not really help the construction or evaluation of alternatives.

The technical group reported after a couple of months' investigation to management that integration would take at least two years. Management was surprised by this - both products were almost ready for release, so why would it take so long time to merge them? The product had to be released in less than half a year's time. The technical group's response was that then one of the products had to be selected and the other one discarded. They did not want to give any advice on which to choose, however, the criteria how to choose would be up to the management to select. Management told the technical group to rework the schedule and produce a solution with at least partial deliveries. Outside the technical group, development had continued in parallel as before.

Then, the decision was made to discard the American system in the long term and market the Swedish system as the product. The reason to choose the Swedish system was that it was more advanced, in the sense containing more innovative solutions. (Much of these included new ways of communicating with other systems, and other types of systems. Enabling intercommunication and information exchange is a strong trend in many domains.) The message to the market was that the products shall be merged into one product family. The internal message was that customers using North American products shall be migrated to the Swedish products when the functionality offered was the same in both products. You have to be very careful with the message to the market, so you do not loose customers. How would you feel if you after you have bought a product get to know that the product will be replaced by another product? There is therefore a difference between product decisions and technical decisions. The same technical decision was made again (Swedish platform and American user interface) was made three times (this was the second).

After this decision, nothing really happened to follow it. The decision was not accompanied by any real means of steering and controlling the organization, such as assigning or not assigning money to different departments. The American system continued development and was released and installed at some customers' sites. The Swedish system also continued development and was released. To the customers, however, the organization acted like a single unit, with two products that in the future would be merged into one. The customers were understandably confused which product to choose and if they would get support on the products they had.

After about two years, the decision already taken was eventually accompanied by economical means of control: no further development money was assigned to the American development department. They were assigned other tasks, and some people had to leave the company. One remaining task is to provide migration solutions to customers that will enable them to "upgrade" to the new version of the system, the Swedish system.

The Swedish system took longer time than expected to become stable enough for large installations. The system configurations were increased in size in steps, and it took about 3 years before it could fully replace systems based on old products.

7.

Both systems' overall architecture was similar – a platform with (similar) subsystems using this platform. Different variants of combining the subsystems from the two systems were considered. Although the functionality was similar, the implementations were very different and difficult to combine. For example, the Swedish system made a fundamental assumption that everything in the system conformed to certain rules (to enable nice integration of the different subsystems), which of course the American parts did not.

The alternatives discussed were not fundamentally different, but were variants of selecting subsystems and using the Swedish platform. To a large extent, this limitation stemmed from the instructions to the group, which was to find an integration solution where the Swedish platform and the American user interface was used.

In the end, no integration solutions were required, since the two products were developed and released in parallel. However, there will be a need for migration solutions in the future, for customers who want to upgrade from their (American) system to a newer version (Swedish system).

8.

The overall goal for the “true” integration discussed (Swedish platform and American user interface) was to enable rapid integration and delivery – which was as said considered not possible by the technical group. The eventual decision, to continue developing the Swedish system and put the American system in maintenance-only mode, was mainly based on the innovations and novelties presented by the Swedish system.

9.

See response to question 6. Integration in the sense of the first decision (use Swedish platform and American user interface) was certainly more complicated than management believed (as estimated by the technical group). It took longer for both systems to be mature enough.

10.

Yes, the Swedish system has been very well received by customers. Due to the economic recession, it took longer time than it otherwise should have done to make money from it. I know too little about the American system to be able to answer.

11.

I. There must be a tough management who can commit to and implement integration.

II. Communication is extremely important, in all directions: management to developers, between managers in the two previously different organizations, etc. (The management in North America and Sweden gave different signals to their respective staff.)

III. Decisions must be accompanied by economic means of control, i.e. assignment or withdrawal of money to make them come true.

12.

The (global) development organization has changed. The Swedish part is relatively unchanged, while the American department has been refocused on other products.

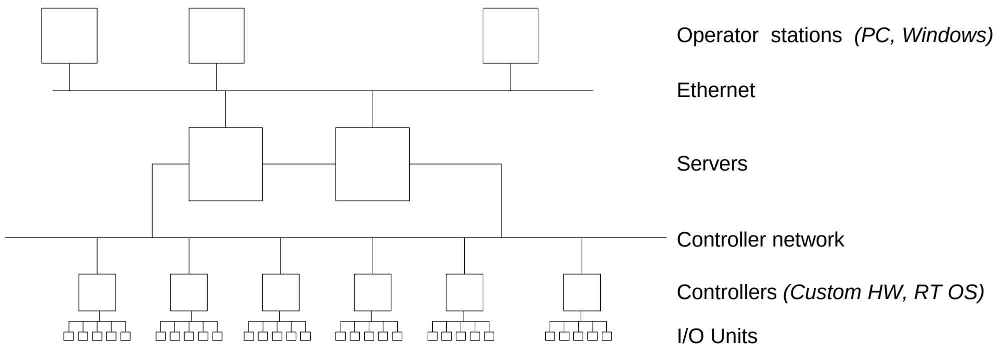
It should be pointed out that the factual sequence of events was not necessarily bad or wrong, even though the decision was changed and a product was released that very soon was transferred to maintenance-only mode (no further development). It is difficult to say what could or would have happened otherwise. And it is not trivial to evaluate the success – short term or long term? Current cash-flow, survival and competitiveness?

Interview I_{Cb}

1 & 2.

The Swedish company acquired the US company in 1999, producing base products. Both companies had started developing the next generation of their respective products around 1997 and the initial goal was to deliver it ca 2000. Both companies had acquired other companies in the same domain during the past decades (and struggled with the same problems).

The overall hardware architecture of the systems can be described in levels: at the lowest level there is a real-time network bus with control units attached to it. The software in these is based on a real-time operating system. Many different manufacturers markets such control units, each based on specially built hardware, and with proprietary communication protocols. Important requirements at this level are safety and reliability. To each control unit, a number of I/O units (sensors etc.) are attached. The main cost for customers is the I/O units. At the level above, there are a number of servers mediating data to operator stations. The operator stations are based on Windows, and the requirements at this level are e.g. that operators must be notified about events. A large part of the interface to the servers, both upwards and downwards, is based on an industry standard (this was true to a less extent for the US system). At this level, the architecture is the same for the two systems of the case.


Operator stations *(PC, Windows)*
Ethernet
Servers
Controller network
Controllers *(Custom HW, RT OS)*
I/O Units

The approximate size of the Swedish system is at the controller level 1-2 MLOC. This includes all support tools built in-house; compilers etc. More than 100 people work with the software at the controller level (there are strict requirements that what is built is correct). The middle (server) and upper level contains approximately 2,5-3 MLOC for the base functionality only (there are ca 30 options that can be added to this). More than 100 people work with the software this level.

These systems were developed during the "dot com boom", so we included some web technology. It did not give any particular extra use, did not generate any extra money, but we have to live with it. We did not see the problems with this technology in a real-time embedded distributed industry environment. The US system built on DCOM, which later proved to be problematic.

For both systems, an important requirement was to be able to integrate with other systems, e.g. a number of different controllers.

3.

The reason was the company merger, and to rationalize development resources. The systems were basically equivalent, so no major new functionality would be added.

4.

All source code completely controlled (apart from the operating systems).

5.

The following stakeholders:

- Business area responsibility managers (responsible for costs and incomes)
- Development managers
- Each country's line organization (fight to keep their jobs)
- Sales and marketing. One responsible in US organization and one in the Swedish.

All conflicts of interest have been between the Swedish side and the US side. Indifferent of roles, people like the system they already know. People want to keep/save their home system and organization. The sellers must promise customers support for the system they sell, and assert that the system being sold is the future.

No individual developers have influenced any decision.

6.

Higher management said: "try to agree, and reuse as much as possible". They did not realize, or did not want to accept the fact that people would start defending their own system, their own jobs, and not being objective. Two problems result from this:

- Progress is slow
- When some decision eventually arrives, relations between people in the two organizations have started to suffer.

We had 1.5 years of endless meetings with a small group, with me (chief architect of the Swedish system) and the chief architect of the US system. We said immediately to each other: "this is stupid; we should just choose one system". During the meetings, no decisive technical differences were found. Both were of good quality, offered the same functionality. They were similar, both the overall architectural structure and choices, and the modules of the systems often provided very similar functionality. This made it easy for us in the group to understand one another at the technical level. It was, given the circumstances, a very good discussion climate. Also, both were in the same life cycle phase (next generation being developed).

Eventually, there came a decision from above, to continue the Swedish system and discontinue the US system. This decision came one year after the US system had been released. It had been much better to make this decision very early after the merger. It could have been done, since basically no new information to base the decision on had arisen during this period. Problems with this delay were:

- The US new generation system was discontinued one year after it was released and introduced to the market. We now have to support this system for some years to come. The decision could have been made before the system was released.
- We lost approximately one year of development efforts by developing in parallel, and also by losing focus from producing the right thing with high quality. Much effort was spent in positioning ourselves towards each other, internally in the company.

The decision was basically not based on any technical factors, but rather:

- Higher management mostly consisted of people with a history from the Swedish organization.
- The Swedish system to a higher extent fulfilled higher management's requirements, their ideas what this type of system should be able to do. This does not necessarily mean that it fulfills the customers' requirements to a higher extent.
- The Swedish system made more profit, while the US system lost market shares. One contributing reason may be that customers saw they were acquired by the Swedish company and did not believe in its future. Also, the US company was offered for sale for a reason.

- The cost for salaries was lowest in Sweden (a relatively minor city compared to the other large cities in US and other countries involved as well).
- The development organization at the Swedish site was the largest. That should make it easier to rationalize, to close some of the smaller centers instead of one large.

The only small technical issue that influenced the decision was the US choice of DCOM. Even Microsoft announced (if you read between the lines) that DCOM was not very good technology, not suited for systems in an industrial setting with high requirements on e.g. fault tolerance. Maybe this made it easier for management to make the decision they made.

If both systems are acceptable, the technical are not decisive. The decision should be based on other factors.

However, some good things were that we who were involved in the small group learned a lot.

It appears to me (but I am only guessing) that it was the very highest management that could not make a decision, and that managers somewhere in the middle felt locked by this, had their hands tied behind their backs.

One thing making decision difficult is that there are no established ways of valuing a software development organization. With a production plant, you have many “hard numbers” to compare.

7 & 8.

As the small group found, very much was fundamentally similar in the systems – the same overall architecture and the same modules existed, for e.g. alarm, redundancy, history, subscription mechanism. These components often provided almost identical functionality.

There were a few differences in technology choices (one of them, choice of DCOM, already mentioned). The Swedish system focused on providing an object model and this was a big difference from the US system which was more functionally oriented.

We can ask the question: why merge/integrate at all? We, the chief architects, thought it was a bad idea from the start. Some problems with the approach:

- Coding conventions, naming conventions. Why mix two worlds?
- If a particular module was to be reused from one system and used in the other, the person receiving it will not be motivated to manage it properly. He would have to leave what he had created and maintain and develop something in another style. Also, the person handing over the module would not be motivated to transfer knowledge (since the reason for handing it over is probably that he is leaving the company).
- There will be a lot of additional work to ensure the quality of the integrated product. (Today, there are 6 months of quality ensuring activities prior to each release.)
- The framework, underlying assumptions of the modules were different: failover, supporting different (natural) languages, and error handling. This makes it difficult to reuse modules. There are no standards at the base – or maybe too many!

If there is no extra functionality to gain, one can well ask the question why try integrating code modules at all. If some modules in one system were very much better than the equivalent modules in the other, it had maybe, possibly been worth it.

Once the decision came that the Swedish system should be further developed and the US system discontinued, some decisions to reuse parts became easier. The user interface code is fairly well separated, and some options at that level were possible to reuse from the US system.

One system communicating with the server level had been developed for the US system. This subsystem was moved from the US system to the Swedish system. This was not too complicated, due to two things:

- The communication was already based on the industry standard, so it was possible to adapt it for the Swedish system (also supporting the standard).
- This system had previously been inherited from an earlier company acquisition, and adapted to the US system. Adapting it once again was easier as most of the issues related to moving this piece of software already had been identified.

This took approximately 2 years for 10 people to do. This relatively modest size made it possible to adapt this system in a controlled manner. Rewriting this system was not an option (for cost reasons).

Within the merged company, there are more than ten different controllers, each with different communication protocols. Support for the new controllers in the servers was added by reusing code from the US system. (This large number of controllers to be supported originates from previous company acquisitions, and perhaps two or three generations of at least the Swedish controller. It is important to support all these old controllers, with old I/O units, since each installation contains so many I/O units (with controllers) being the large part of the cost of an installation.)

The choice of reusing these parts in this way was based on the directive to transfer the functionality in the cheapest way and within reasonable time (ca 2 years). These solutions were not the technically most elegant. Also, another motivation for reusing the code is that we are keeping US people to be able to support the installations of their released system, so why not let these people also work with their own code?

9.

Different interests have been described.

We lost approximately one year during the discussions. Although the hundreds of developers were not directly involved in the discussions, they knew what was going on and had less focus on results and more on worrying about the future of their system and their jobs. There were more focus on exceeding and outdoing the other system than really providing a stable high-quality product for end customers.

10.

Yes, today the Swedish system is very successful. We have won one award for the best product within our domain. The people who have worked with it are very proud.

11.

Most important:

- Higher management must provide clear information and directives. If you as a developer are assigned other tasks, or your system is discontinued, you will complain for a week and then adapt to the circumstances. It is much worse, and unproductive, to live in a long period of not knowing.
- It is more important with a clear decision than a “totally right” decision. You only need to make sure it is not totally wrong. In our case, none of the two systems had any obvious advantage, so choosing any of the two systems immediately had been much better than waiting so long for a decision.
- Keeping the organization motivated.

12.

I believe the next time we will do the same mistakes. We did the same mistakes 15 years ago when we acquired another company, but almost no one is left from this time. It seems as each generation of managers needs to do their own mistakes…

Interview I_{Da}

1 & 2.

A company merger between a North American company and a European company with one department in Sweden (where the interviewee is located). The systems are major systems for the core business of the companies, with the same basic functionality. These systems, as well as several other systems from other companies in the same business domain, have a common ancestry, as companies and divisions and systems have been sold and merged throughout the past decades. Therefore, the basic architectures of the two systems (the North American and the European) are similar, and many concepts are defined and used in very similar ways, although they have diverged.

The American system has been developed in sales projects, while the European system has been developed by a development organization, which has had a longer-term plan for the system, and been able to transfer all changes made during sales project to the common artifact. During recent years, the American system had been profitable while the European system has not. Funny, but now the opposite is true, i.e. the Swedish system makes profit and the American makes a loss! Despite being advanced, of good quality, and accepted by the users, it has had too small customer base to really support the costly development. This is true for all actors on our market, i.e. the market is too small to allow for development of advanced systems at a small cost per system. For the American system, only little money had been spent on modernizing and evolving it – which is one reason it had been possible to make profit during a series of years. And the development now being done is perhaps the reason for their current losses!

At the time of the company merger however, the American system started to loose its customers, who considered it to be aged ant not modern enough. Two major improvements had to be made for the American system: a new HMI (Human-Machine Interface) and a new data engineering tool. The list with development needs has grown longer since then. The European HMI was chosen based on market considerations, but a commercial data GIS tool was chosen as basis for a new DE-tool instead of the one being part of the European system. The reasons for this were several: 1) Higher management wants commercial software, 2) there was a will to use the standard IEC 61970-301 (CIM), and 3) there was one particular important use case not supported by the European data engineering tool (to generate graphics from model data).

After the merger, the European system and the American system has been considered two separate tracks which will be maintained and evolved to keep existing customers. There is an ongoing discussion which components can be common for both systems.

3.

Upper management wants to integrate in the long term, but this is difficult and requires large investments. Assignment of development money in delivery projects does not follow this strategy. Projects including costly project specific development are not punished. The laws of market economy are disabled.

The reason for the American department to choose the Swedish HMI was based on market reasons – it does not require any extra hardware but runs on a standard PC.

4.

No legal restrictions – all code developed within the enterprise.

5.

[Not directly discussed]

6.

The Americans made their choices, and the Europeans made theirs. At a high level, the choice of system components to integrate into the American system was based on needs from the market.

7.

Previously, the European HMI had been running in a server environment with a dumb client. As the new PC-based HMI was developed, the existing protocol was used and ported to CORBA. This was not entirely a good design choice, since the protocol was built assuming that much of the intelligence lay on the server. When more and more intelligence was moved to the PC software, one should have removed the same functionality from the server. This has not been done, and therefore functionality is today duplicated (both server and client), and the protocol is unnecessarily complex.

When the American system should use the European HMI, the same communication protocol was to be used, so it was the American server software that had to be modified in order to make communication work. As they had a common ancestry long ago, it was not too difficult to map the concepts used in the American server and the European HMI to each other.

It was such a large project that it would require too much effort to do all that "should" have been done.

8.

[Not directly discussed]

9.

The project was a little late, somewhere between 6 months to a year. There were two parts: 1) development on the Windows side, and 2) on the server side. I think the Windows part followed their plan while the server side took longer time than planned. Interviewee I_{Db} should know how much.

10.

I think the new HMI has been received well on the American market. Also here interviewee I_{Db} should be able to contribute with more info since he has the contact with the Americans. But as said, the costs have been large and the American unit currently makes losses.

11.

Defining an interface so that a change on one side will not affect the other. Interface can be seen on at least two levels: first, a data model (UML, IDL, XML), and second, an infrastructure (e.g. CORBA, SOAP, file system, SQL database, middleware). Also, the protocol is an interface.

Also, the most important asset for a company is the staff, the humans who know the systems. One should be anxious about the staff. The source code is worth little if the personnel leave.

12.

The American division has the overall product responsibility (for their product, i.e. the American server and the Swedish HMI). Sweden evolves the HMI as a subcontractor for US organization, and to evolve their own product (Swedish server + HMI). We now have a common product and the long-term goal is that all components in the product will be common. The concern has had this goal for a long time but it has been difficult to reach there, partly because of the issues discussed above but also because of different requirements.

Interview I_{Db}

1 & 2.

Previously two competing companies, now part of the same concern but there is still some amount of internal competition. One North American and one Swedish company/system. Large systems, the core software of the business. The American system was more successful, but not necessarily technically supreme.

The Swedish HMI runs under Windows, while the US HMI has been server-based, using X windows. The complete Swedish system (all server parts and the HMI) was approximately 1 MLOC.

The US organization has traditionally been very focused on sales projects. The Swedish system has undergone an integration similar to the current situation more than a decade ago, and therefore the system is a little more generalized; there are also staff assigned to maintain the system.

3.

The reason for the overall goal – one single system in US and Sweden – was to rationalize development efforts. The reason for the integration that actually has taken place so far (se rest of interview) was that the X windows based solution suddenly was hopelessly out of fashion, and a modern HMI was needed. They would have developed their new if they had not chosen to use the Swedish HMI instead.

4.

The systems are owned by the respective organizations (US & Swedish) – no out-sourced or third-party (apart from fundamental support technologies). The US organization has however been reluctant to show or give away their source code and documentation. The Swedish organization consciously decided that if integration is to succeed, we must show generosity, and sent all their assets (source code etc.) to the US staff.

5.

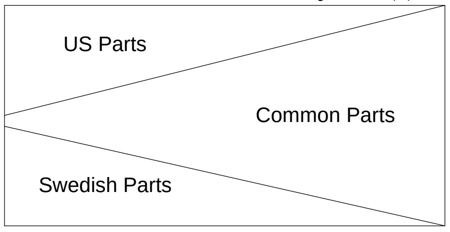
Similar customers, although the US system had mainly US customers, and Swedish system had customers elsewhere in the world.

Higher management responsible for both systems (wanted integration in the long term).

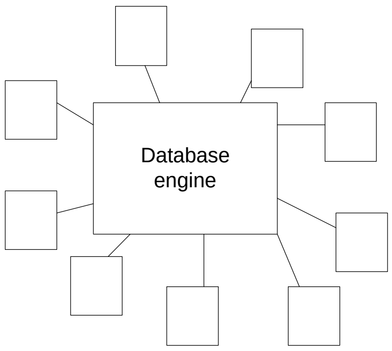
Otherwise, similar stakeholders.

6.

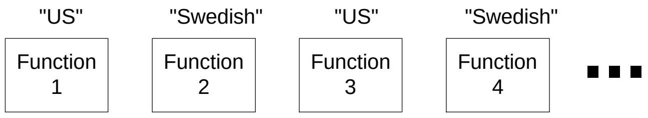
There was a decision 3-4 years ago to migrate towards a common system, so that the parts specific to the North American system or the Swedish would become less and less, and the number of common parts would increase until the whole of the system(s) were common:



The systems could relatively easily be conceptually divided into functional parts. In the minds of the people involved, there was one part that was central, the database engine, and all other parts surrounded it:



Dividing the system conceptually in this way was easy. The difficult part was to decide which part to pick from which existing system. The overall approach would then be to pick each of these parts from either the American or the Swedish system:



The Swedish staff decided that building trust was essential. Swedish organization is exposed to a risk, and the US organization also. Cooperation without hidden agendas is the most beneficial for both parties. The Swedish staff sent their complete source code to the US development organization on day one.

There were three major conditions that made integration of the Swedish HMI into the US system successful:

I. Cooperation. Many problems, pro and con, were debated among the Swedish development organization, and eventually they decided they wanted it to happen – otherwise it is easy enough to throw a spanner in the works. Issues were: if the US system gets the Swedish HMI, they will become a threat to the Swedish system in the long term. On the other hand, the American market will have become used to the exact same user interface as the Swedish system, which is an advantage. And what if the US organization will develop their own new, modern HMI – that will mean competition with the Swedish system. Would it be disloyal to the Swedish server staff to help the US system? Also, it would not be fair play to consciously work against a higher management decision (to in the long run make the systems converge and more and more use the same components). All in all, the Swedish staff decided they had to be positive and generous and make an effort to make integration happen.

II. Similar requirements. A demo of (offline part of) the Swedish HMI were made. It was shown that many of the fundamental requirements from customers were already implemented – which was to a large extent because the requirements for the Swedish HMI and the new US HMI were written by the same (US) consultants. Objects in the user interface had the same names, meant the same things, and were structured in the same way in (the requirements for) both the American and the Swedish system. "Children of the same spirit." The American staff was convinced that integration was both possible and cost efficient. If things had been more different they would probably not have chosen the Swedish HMI.

III. Technology. See next question.

7.

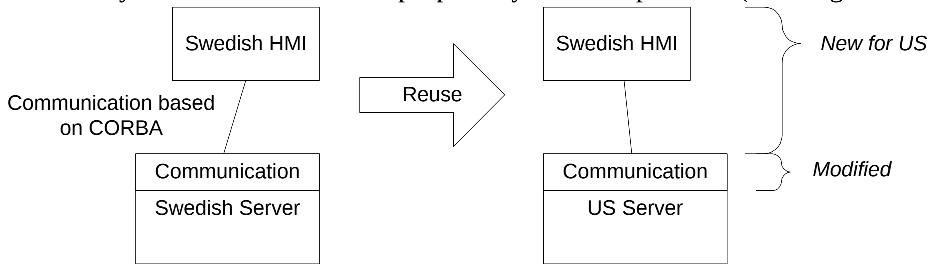
Previously, the Swedish HMI was built as three communicating processes, each written in C in a monolithic manner. There were flags and conditions spread throughout the system, and a maintainer had to understand the complete system with all variants encoded by these flags and conditions. As a solution, five years ago the HMI was decomposed. When studying literature on object orientation, a component approach was chosen to deal with some of the potential problems with OO. That is: keep components apart, define interfaces that you honor and do not break. The system was moreover designed within the component technology as a platform and "ordinary" components implementing certain generic interfaces, mandated by the platform. From the view of the platform, all components are the same and can be easily plugged in and out. The platform contains 10-20% of the source code.

If the old monolithic system had been in place when the discussions on integration had started, integration would not have happened. But due to the componentized architecture, it has been straightforward to implement new functionality, either as new components, or in some cases by extending some components. (There are also other experienced benefits: there is a way to retire code simply by plugging out a component – in the old system it would be expensive to go through all source code and remove some if-blocks, only to remove functionality; it is a fast way to find an error, by removing one component at a time and reproducing the events leading to the failure; it also enables new license models, as it becomes possible to sell only some functionality.)

Communication with the server, either the US or the Swedish server, is implemented in each component that needs it – there is no communication implemented in the HMI platform.

8.

With the component architecture already in place, this was the way to do it. 95% of the communication to host in the US system uses the Swedish proprietary IDL and protocol (building on CORBA).



9.

The integration has proceeded approximately according to the time plan, and integration is complete.

In order to make the US server communicate properly with the Swedish HMI, the US server software has had to be modified. This made customer deliveries late, and in such cases it is convenient to blame the subcontractor.

10.

Internally, yes (do not know about customers to US system). The American organization is satisfied, and has outsourced some other software development to Sweden, which can be seen as a receipt that they are satisfied with the HMI and the service.

11.

Success factor: Building trust is absolutely essential. People are reluctant to distant cooperation with new partners unless they see a real need. Sweden made great efforts to show they had no hidden agenda, such as sending all the source code on day one. (US may even consider Swedish staff to be naïve and a little stupid, but it is on our treat)

Also, a certain amount of “luck”, or “timing” is required. At the same point in time, the three conditions listed under question 6 were fulfilled. If one of them had not, the integration had never taken place.

Pitfall: to not have thought through the issues completely. Also, difficulty to cooperate across the Atlantic, for example giving support via phone: listening to someone’s description of what is visible on screen instead of seeing it yourself.

12.

As described, the component architecture that was already in place has been even more emphasized.

The US now has the product responsibility, but they have not prioritized integration in the sense of the overall goal, only one system.

Interview I_{E1}

1 & 2.

A note on terminology: the SILVIA system is usually talked about in terms of “models”, meaning mathematical description of different physical systems.

Before the SILVIA system, FOA had models at the system evaluation level , and the industry (SAAB, Bofors, and Ericsson) had models on a more detailed level. They were written in different programming languages and reflected different philosophies.

The Swedish defence forces were planning to develop of a new medium range air defence missile system and wanted to be able to evaluate the effectiveness of different solutions early in the development process. After several discussions between the armed forces, FOA and the industry, it was decided to develop a family of simulation models which could fulfil this task.

At a meeting in 1982 (see D_{E1}, [28]) it was decided that this system (to be named SILVIA) should be built, and responsibilities were assigned to the different parties. Both FOA and the industry saw a value in cooperating and build a modular system where both the level of detail could be chosen.

3.

There was a new need for models, and they were supposed to be on 3 different levels of detail:

A. System evaluation on an operational level with a complete air defence company consisting of several firing units and surveillance sensors in different terrains, threat scenarios and tactical situations. The purpose of this level of detail would be to evaluate the performance of a complete antiaircraft company during an attack, to be used by military leaders. On this level the importance of main air defence characteristics can be evaluated e.g. number of firing units, missile range, system reaction time, number and quality of surveillance sensors etc.

B. Firing unit level. This level would contain local surveillance sensor, missile launcher with missiles and fire control. The purpose of this level would be to be used during development of new technology (e.g. evaluating different missile guidance technology).

C. Subsystem level. This level would contain e.g. target seeker or missile guidance and control system ("autopilot"). This level would also be used mainly by industry to simulate the subsystems during development of new technology (new subsystems). An issue with this level was industrial secrecy – the industries did not want to show these models to any external industry, nor to FOA.

The same entities are modeled at the different levels, only with different levels of detail: level A would contain simple models of all entities, level B a detailed level, and C a very high level of detail (subsystems of the entities). It should be possible to “lift” the models one level and choose a more or less detailed variant of a model.

Interfaces are specified, so the models themselves can be plugged in and out. In this way, the industry does not need to ship their C-level models to someone else. Simulations intended to evaluate the systems at the detailed subsystem level of detail (level C) would be carried out at the industry, while FOA and others would have B-level models available for e.g. system evaluation.

To be able to integrate the different models, a common “environment” had to be defined. This included three main parts: the model of the atmosphere, the model of the terrain, and generic mathematical functions.

4.

All source code was available within the cooperation project except for code on the subsystem level. The code was owned by the developing organization (e.g. FOI, Bofors). All code was developed in house. At the time when the development started there was no suitable, Ada-code available, neither commercially or open source.

Ada was very new, we even had to develop a package for mathematical functions and coordinate transformations ourselves

5.

FMV (“Försvarets materielverk”, Swedish Defence Materiel Administration) invested money in the project.
Industry saw a need for a common framework in which to build and execute their models.
FOA saw the need for system evaluation.

6.

Most of the existing models were written in FORTRAN. But the previous system evaluation model (AKYTA) developed at FOA was implemented in SIMULA and it would very hard to integrate them. With FORTRAN, it is possible to solve problem that are mathematically complex, but not with a high level of system complexity. FOA knew quite a lot about the advantages of object oriented design through the use of SIMULA and Ada was an emerging technology in that field supported by the US Defence.

With Ada there was a good possibility to create reusable code and it was decided that everything should be written from scratch, in Ada (see motivation below).

There were no estimations of cost to support this decision, neither explicit nor in the heads of the members.

7.

The new system would be built in Ada (see motivation below), but none of the existing models were written in Ada. Everything needed to be built from scratch, reusing the mathematical models.

Ada was developed to make it possible to construct very large complex computer models in a modular fashion. Ada is a strongly typed language and built on the ideas of object oriented design. The development can be divided in two phases, specification and implementation. During the specification phase the main object types and operations on these are identified. These are collected in package specifications consisting of type and function declarations somewhat like UML. This technique was used in the beginning of the development and was excellent method to discuss ideas and implementation issues in a structured and precise way.

The interviewee made a pre-specification of Ada interfaces (FOA had the integration responsibility), and then the industry was involved again. This specification consisted of the subdivision of the code in functional modules (packages in Ada), simulation entities (tasks in Ada) and objects (mainly records and arrays). Ada allows you to compile the specifications and check for inconsistencies early in the development work, which was very convenient. It is also quite easy to refine the data structures without changing the over all design.

Many of the ideas from earlier systems were reused. It was quite easy to transform SIMULA-code to Ada as both languages are object oriented.

8.

The Swedish defense had a great like for the Ada programming language, and it was becoming a standard for embedded software in military systems. Ada is a strictly typed, object oriented language with excellent facilities for exception handling, making it possible to develop robust code of very complex systems. The task feature in Ada makes it quite easy to write event driven simulation models.

9.

There were no advanced project activity schedules. It was a very stimulating project to work in. FMV (“Försvarets materielverk”, Swedish Defence Materiel Administration) invested money in the project, and the project had reporting obligations towards FMV. The interviewee is surprised how patient they were at FMV.

One year was spent in developing and modifying the interface specifications and models (Swedish word “tröskande” was used), before actual implementation could start and be carried out in parallel in separate places. It was not frustrating, but took longer time than expected (see D_{E1}, [28]). Even when implementation started, there were close contacts, mostly at an informal basis.

There were lively technical discussions, but in the end, the expertise within a certain area had the final say. For example, Ericsson was the expert on radar systems. There was a will to compromise, and there was an apparent lack of protecting one’s own territory (Swedish: “revirpinkande”). The project was quite a success. There were several reasons for this:

- A patient and understanding customer (FMV) who did not interfere with development work more than necessary.
- The participants were technicians and appealed by the technical problem, everyone wanted to do a good job.
- There was a respect for each other’s competence.
- There was a common project group.

Surprisingly little was changed in the interface specifications after the first year. This can be attributed to previous experience with development projects. The participants succeeded in resisting the temptation to start coding too early. Some things are possible to implement in only one way (maths, data structures like vectors and matrices) and those could be implemented immediately.

Some Ada features were not completely known. Especially “tasks” was a cause of problems and concern: concurrent programs are a hassle to debug, and there were deadlocks. It is easy to get enchanted by the elegant school book examples. It was however impossible to back (1,5 years) and remodel this. It is actually elegant with separate units of processing (tasks, processes), but it becomes very important to understand the communication channels.

It took much longer than expected. After perhaps 2 years there was “something” executing, not ready for production use (but might remember incorrectly).

The only frustrating part was the discussions on ownership and usage rights. FMV paid, at the same time as the industry would be allowed to use it. The legal experts, with the interviewee as a spectator, debated 1,5 years before all managers had signed all papers.

10.

In terms of the functions offered by the system, SILVIA fulfills the expectations: simulation, graphics. The input data structures are however complicated, and in practice, the only users have been the developers (and their successors). Possibly, the air defense has run SILVIA on their own. It has been executed at FOA, Bofors, and Ericsson.

11.

The factor contributing the most to the success was the enthusiasm, lively discussions and fun people involved had, and the lack of strict management of the development process. Also, it was important that all involved understood the purpose of the system.

12.

Many of the packages developed in the SILVIA project has been reused in later projects. Unfortunately Ada is loosing steam and C++ is in the limelight (a complex and inferior language in the interviewee’s opinion)

Interview I_{E2}

1 & 2.

The history from 1988, when the interviewee became involved in SILVIA, includes:

- Integrating new modules from industry (FOA had the integration responsibility), including verifying and testing them.
- Some new functionality was introduced, like sensors and functions disturbing sensors.
- Error removal.
- Port to Sun. Earlier, input and output data was handled on PC, and had to be moved to and from the VAX computer manually by carrying magnetic tapes between rooms. The port enabled all data to be processed on Sun. However, the industrial partners were lost by this port, partly because it was difficult to find new staff functioning as successors.

There were a number of versions and special versions released in a hurry. Rapid results were requested. The code was cleaned afterwards – sometimes this took a while.

1994 the interviewee took over as project leader for SILVIA. 1994 there was a change in project financing, from grants from the Swedish defense to a small part grants and the main part based on orders. The volume of development decreased, from several persons to two fulltime staff plus some degree projects every once in a while.

There were two other systems used for similar simulations: FBSIM (Swedish: “förbandssimulering”, simulation of units) and LvTASS (Swedish: “Taktisk analyssimulator för luftvärn”, tactical analysis simulator for air defense). FBSIM was used to simulate units (Swedish: “förband”), as SILVIA, but with a large portion of tactics and little technology (SILVIA can be described as large part technology and a smaller part of tactics). FBSIM was also intended to simulate not only air defense but also army units, and “everything”. LvTASS is an air defense adjusted planning analysis tool, used by the air force to plan attacks in the presence of air defense.

There was a major initiative within the Swedish defense called IT4, which financed IT related efforts, among them FBSIM. FBSIM, as well as SILVIA, is written in Ada.

3.

The systems simulated similar things, and to some extent the same things. The user base for all systems was small.

4.

All source code for all systems was owned by different parts of the Swedish defense, and was accessible and available when needed. It was FMV (“Försvarets materielverk, Swedish Defence Materiel Administration) who ordered the investigation whether and how to integrate, and it was assumed that there would be no legal obstacles concerning accessibility, ownership etc. The exception was the level C models which were maintained and used only within the industrial corporation that created them.

5.

[Not directly discussed]

6.

The opinion of the FBSIM project was that the technology part of SILVIA should be incorporated into FBSIM. Why this was not done was because the SILVIA project could not be disturbed, it did not have total control of its own development.

An external investigator studied the systems (see D_{E2a} [1]). There were lots of discussions. In the end, FMV chose to retire LvTASS. There was no replacement for the functionality offered. FBSIM was only used at one site, and was in practice almost retired. The FBSIM GUI was however useful and was used in SILVIA. There have been discussions about running SILVIA within the Swedish Air Defense (Swedish: “Flygvapnet”), where LvTASS was used previously, but so far this has not happened – only a few times when the interviewee has run SILVIA together with the Swedish Air Defense.

The cost of maintaining several systems was also one reason to decrease the number of systems.

7.

Most of the SILVIA GUI is now identical (same source code) as for FBSIM. That the other functionality of the retired systems was not transferred to SILVIA has several explanations:

- The Swedish Air Defense had a slight attitude that IT4 had pushed a model upon them they had never asked for.
- One central person (developer) moved to a consultant company.
- All systems had expectations set higher than could be met.
- There has been an over-belief in simulation models: “simulate instead of thinking”.

Both FBSIM and SILVIA are displaying flight paths and other elements with a map as a background. Both also model the same narrow domain (air defence), and as said are both written in Ada. There was a desire to make them look identical, including not only the graphics but also the position of menus, buttons etc. At least SILVIA is loosely connected to the GUI; the GUI basically calls the SILVIA executable, which reads input data from a set of text files. Considering all similarities and the loose connection between GUI and simulation model, reusing the FBSIM GUI in SILVIA should be relatively straightforward (with little modifications). The task was outsourced to the same consultant company that had written the FBSIM GUI. However, the task was more complicated and took longer than expected. For example, flight paths are not modeled in the same way in the two models, and the log files and input data format are different between FBSIM and SILVIA.

Clarifications made by the interviewee after the interview, after the feedback loop, in Swedish:

Svårigheten som konsultfirman framför allt råkade ut för var nog (jag kan egentligen bara gissa) mängden indata till SILVIA och även starka krav på hur vissa indatafiler var uppbyggda samt flygbanornas representation i SILVIA.

Det underlag som konsultfirman hade var en spec i form av en användarhandledning till det blivande GUI (skriven av en fd PL för FBSIM), en indatabeskrivning och att delar av konsultfirman tidigare arbetat med SILVIA i dess barndom.

Jag skulle vilja säga att det GUI vi fick inte riktigt fyllde våra förhoppningar och FMV har i efterhand fått lägga några tilläggsbeställningar för att vi skall kunna använda det och än idag så finns det vissa delar kvar som inte är bra.

Vi kommer få en tilläggsbeställning för färdigställande till sommaren.

8.

There was a requirement for a new GUI to resemble FBSIM, so it was natural to reuse the FBSIM GUI.

9.

[Not directly discussed]

10.

[Not directly discussed]

11.

Prerequisites: Close cooperation, not too many people, to know each other personally.

12.

The lesson learned is how to use SILVIA, which problems the SILVIA model can solve. There have been no new functions for a long time.

13. The interviewee's own notes, after the feedback loop (in Swedish).

Eftersom det skett flera modellval så ska jag försöka att reda ut dem för dig. Det sker ständiga modellval beroende på vilka problem som är aktuella. När det gäller SILVIA så är det framför allt valet mellan SILVIA, FBSIM och LvTASS samt det senare valet om SILVIA skulle inkorporeras i FBSIM. Runt dessa modeller finns andra som vi inte ens nämnt men som komplicerar bilden.

Om jag minns resonemanget kring modellvalet så stod det tidigt klart att SILVIA skulle behållas eftersom ingen av de andra två hade den detaljeringsgrad som SILVIA hade på luftvärnssystemen och det fanns (finns) behov av denna detaljeringsgrad. I valet mellan FBSIM och LvTASS så var det däremot lite av en äpplen- och päronjämförelse.

LvTASS gav bäst uppfattning om attackens möjligheter i ett scenario eftersom verktyget fokuserade på detta. Däremot kunde man skjuta ner månen med RBS70 (ett korträckviddigt luftvärnssystem med ca 5 km räckvidd), dvs. luftvärnet kände inte igen sina egna system. Systemet körde på mycket speciell konfiguration av HP-dator och hela systemet var hemligt. Luftvärnet var helt beroende av flygvapnets utvecklingsbehov och hade mycket litet inflytande på modellen.

FBSIM simulerade även andra delar av arméförband (fast jag minns inte aktuell status vid utredningen). Luftvärnet hade kontroll på modellutvecklingen och var inte beroende av andras utvecklingsbehov. Däremot var attacken betydligt enklare uppbyggd än LvTASS. Datorsystem var Sun/UNIX och portering skedde till PC. Modellen var öppen. Därmed var modellen tillgänglig för fler.

Alla tre systemen krävde en hög tröskel för att börja användas. Försvaret är uppbyggt på att man byter tjänst minst vartannat år och därmed krävs ständig upplärning. Min uppfattning är att detta tillsammans med bristen på inflytande på LVTASS modellutveckling och att modellen var hemlig var det som egentligen avgjorde valet.

Motiveringen till det slutliga modellvalet var att SILVIA skulle användas av få personer för teknisk utvärdering medan FBSIM skulle användas på bredden för taktikutveckling. Stridsträningsanläggningen som skulle byggas i Norrtälje (byggdes senare upp i Halmstad) skulle inte räcka till för alla förband och därmed behövdes FBSIM som ett komplement. LvTASS lades ner i brist på tid för upplärning och i brist på pengar.

FBSIM har sedan dess fått mer arméförband och behållit sin detaljeringsnivå. Det fanns en diskussion om SILVIAs mer detaljerade modeller skulle inkorporeras i FBSIM, men valet blev att låta bli och jag tror att det var min erfarenhetsrapport som bidrog till detta val, dvs. man valde att inte koppla ihop för många detaljerade modeller eftersom man ändå inte kommer kunna analysera dessa situationer. Antingen kör man en begränsad teknisk utvärdering med SILVIA eller ett taktikscenario med FBSIMs enklare modeller. En bidragande orsak var säkerligen att vid tekniska utvärderingar vill man ha FOI med och FBSIM är en modell som vi inte har full kontroll på. FOI är vid användning av en modell både leverantör av resultat och mottagare av resultatet. Ett konsultföretag saknar insikt i problemformuleringen och är enbart leverantör. Ytterligare en faktor var att man inte ville hamna i gapet mellan modellutvecklingar och stå med en halvfärdig SILVIA och en halvfärdig FBSIM. Däremot valde man att välja samma GUI för att dela utvecklingskostnader och för att underlätta att luftvärnet skall kunna använda SILVIA på egen hand. Båda modellerna bibehölls.

Modellvalet gjordes inte av FMV men däremot skedde utredningar på uppdrag av FMV. Valet skedde snarare av luftvärnets studiegrupp där FOI och FMV ingår och utredningen ingick som ett underlag. Dessutom ingick utredaren i denna studiegrupp (och jag).

I dagsläget finns bara ett luftvärnsregemente och därför har man inte idag samma behov av FBSIM för taktisk utveckling. Stridsträningsanläggningen STA/Lv används för taktisk utveckling, utbildning och övningar. Pengar för utveckling är idag ännu mindre tillgängliga och i dagsläget saknas medel för både SILVIA och FBSIM. SILVIA klarar sig på redan uppbyggd kunskap och används som den är. FBSIM utvecklas genom konsultföretag och vid behov genom andra delar av armén men inte av luftvärnet. Modellen är tillgänglig och fortfarande möjlig att användas även för våra problem.

Luftvärnet och FOI har idag bra insikt i vad SILVIA kan användas till och i dagsläget används nog FLAMES i de scenarier som man tidigare tänkt använda sig av FBSIM i när det gäller mer övergripande scenarier. FLAMES opereras av FOI. FLAMES är ett kommersiellt ramverktyg för modellutveckling. Däremot har planerna på att luftvärnarna skall börja använda SILVIA ännu inte förverkligats. Jag tror att den största anledningen är att tröskeln att börja använda SILVIA är hög och att man byter folk hela tiden och att det finns IT-restriktioner inom försvarsmakten som är hindrande. När behov att använda SILVIA kommer är tidplanen pressad vilket gör att även vi haft svårt att lära upp någon. Det går fortare att göra körningarna än att lära upp någon annan och körningarna byggs upp av problemställningen och on-line analyser.

Val av modell sker efter den problemställning vi ställs inför och därför fladdrar modellval. En av orsakerna till valet av FLAMES var tillgängligheten av kunskap inom studiegruppen och därmed tillgängligheten av möjlighet att tolka resultaten genom vår möjlighet att välja scenario och problem som passar modellvalet. Vid valet av värderingsmodell så sker en process mellan problem och modell som är oerhört viktig för att få ut användbara resultat. Denna process är svår att genomföra med FBSIM eftersom det ett flertal gånger har klickat i kommunikationen mellan problemägare och modellutvecklare.

Några diskussioner att använda SILVIA för flygvapnets del har inte förts. Flygvapnet har fortsatt sin modellutveckling utan att snegla på luftvärnets val. Flygvapnet har däremot lagt till ett bättre modellerat luftvärn i sin träningsanläggning (där FOI gör robotmodellerna) och det är snarare vi som tycker att modellutvecklingen borde samordnas. Marina behovet av luftvärnsmodellering ser f.n. ut att gå på ett eget spår. Vi utvärderar för tillfället en modell från Australien för deras räkning. All denna modellutveckling av luftvärn borde samordnas med tanke på skriande resursbrist men sker idag suboptimerat därför att pengarna kommer från olika källor. Redan vid start av SILVIA 1983 så borde luftvärnets och flygvapnets modellering ha samordnats (fast då hade luftvärnet sannolikt inte haft någon användbar modell idag).

Interview I_{F1a}

1 & 2.

See also other resources.

There is one large Swedish information system managing the simulations performed by the 3D simulator. In the US, there are two smaller systems covering some part of this functionality. However, these systems address somewhat different needs, so from a certain point of view one can say that the US site has no system similar to the Swedish system. Perhaps there is some overlap with some “downstream” codes in the US, i.e. run after the 3D simulator. In the US there were methods on paper corresponding to the tasks automated by the Swedish system. Also, the US 3D simulator contains some automations that have been implemented in the Swedish information system instead of the Swedish simulator. There are also some differences between type “A” and type “B” simulations that matters here.

3.

During autumn 2002, a decision was made to merge the systems. *See also other resources.*

However, after that, nothing more happened. Suddenly, we got the request from US users: “What would it take to use the Swedish system with our 3D simulator for type ‘B’ simulations?” So, the reason to “integrate” was that the Swedish system presented functionality for type “A” simulations the US site did not have a corresponding system for.

4.

See other resources.

5.

See other resources.

6.

See also other resources.

A set of activities needed have been defined in a company internal document. The major issue is a common data model. Also, the current proprietary object-oriented database is insufficient, and a common storage format is also being developed. The US 3D simulator will use this new storage format fairly soon, and the Swedish information system a little later.

A number of telephone meetings have been held. The result was that there are too large differences between the current data models (for type “A” in the Swedish system and type “B” in the US 3D simulator). We will try again though, with the particular goal of identifying what in the Swedish information system’s data model is used by the system itself (and what is a model of something external to it). After this, we might make a definitive decision.

The goal of a single 3D simulator is a clear goal, but is far away into the future. Case F2 is one small part of this goal.

7.

We will extend the current system to handle type “B” data and calculations.

Part of the support for type “B” simulations have been implemented in the Swedish information system, partly for demo purposes (which led to the enthusiasm in the US), but only part of the code should be considered prototype, this is an important step towards full support.

8.

The current system has been used for almost ten years and works well. Of course it could be implemented in a better way but the code base is large (> 500kLOC) and it would be very expensive to rewrite it completely. Parts of the system, e.g. the database, will be re-implemented though.

9.

See also other resources.

The part of the support for type "B" simulations that have been implemented in the Swedish system have been according to plan. But most is left to be done, and is complicated by the fact that there are today two different 3D simulators but they will be merged in parallel with the present extension for supporting type "B" simulations.

10.

So far, the steps taken have been well received.

11.

(Same answer as for case F2.) Most important is to sit together physically. Periodical visits, lasting one or two weeks, are required. For critical tasks, e.g. high-level design, longer periods of working together are needed, maybe one month. One cannot rely on tele- and video-conferencing.

12.

(Similar answer as for case F2.) No, not really. Simultaneously and independently, new processes have been adopted from the US site (as a result of the merger). This project has produced all required (very much) documentation.

Interview I_{F1b}

1 & 2.

Large 3D simulators in both US and Sweden, developed and maintained for many years. A Swedish data management system, in use ca 10 years, built around the Swedish simulator.

The project I have been involved in the past 1,5 years is to solve the immediate needs of the US 3D simulator.

3.

Actually, one can doubt whether integration should be a goal. There are several important differences:

- Differences between type “A” and “B” simulations
- Different methodologies in the use of the simulators
- Differences in how customers use the code

So, what would be the value with integration? For users? For the company? A big system that covers everything is more difficult than maintaining smaller separate systems – data interchange must occur at a very well-defined integration point (which according to me should be a common data format and libraries for many languages to access this format).

In an ideal world, you could solve this in a big system with different configuration options etc., but in reality we do not have the resources for this. The Swedish system is more all-encompassing, and works well for its purposes, but it is dubious if we should continue with that path to include more and more. We need some cost-benefit analysis – should we do it or not?

4.

Everything owned in-house.

5.

Sponsoring manager. He is a central physics expert with the task of coordinating all integration efforts. He focuses on long-term integration. He wanted to use this project to probe the possibility of using the Swedish data management system in US. His main concern is to not diverge from that path. Actually, he is not so happy with what I have been doing.

Project manager. The project manager representing users has had the focus on how to help the users, on a more immediate time-scale.

Customer. The main customer for my project is the US 3D simulator team, with their immediate needs (new data storage). They are happy with my project, but currently do not have resources to be able to use it. They expect waterfall-like development, which I cannot do – that is development for management. I want to deliver stepwise.

Line manager.

Swedish people. I have been talking with interviewee I_{F1a} about the interface towards the Swedish 3D simulator and the Swedish data management system.

Users. Users are represented, mainly through one specific person.

Software process people. Software process people are not so happy, they want to impose a waterfall model. This does not work as we have a distributed team with resources both here in the US, in Argentina, and in China.

6.

I have been involved in many discussions, with the purpose of defining different roads towards integration.

The project I have been involved in the past 1,5 years is to solve the immediate needs of the US 3D simulator. A new methodology will be implemented, and the old storage file format will not work. A secondary goal is to also support a future merge of the Swedish and the US 3D simulators. We will create a platform independent storage, built on XML and HDF.

In the beginning, a little more than a year ago, the scope was much larger: to create a unified data architecture, a super data management system. For about 6 months, I had to figure out what was needed, scoping work, and the scope was reduced to only replace the current file format. Then the scope was extended a little, when the people from case F2 visited us in the US. The project would provide data storage and surrounding utilities (the 3D simulator only concerns the physics part). The intention was good, but the US 3D simulator team was not in line with schedule. We had to wait for them. It became apparent that although they wanted the new data storage we would provide, they would not have time to incorporate it and use it. They are not ready yet, and the project is on hold until the requirements are clear.

We have had some disagreement between US and Sweden. The Swedish people want to go straight towards an integration by extending and/or modifying the Swedish data management system. My concern is the immediate needs of the users. The Swedish system is an all-or-none system, and the data storage currently used is not suitable for our needs.

My view of the process is that you should build small utilities to satisfy the immediate needs of the users. At the same time, we should try to reuse as much as possible from the Swedish site. The goal is not to migrate, but to satisfy users' needs. You always have to take into account the time pressure and the pressure from users.

You need a high-level vision, but not in terms of a system blueprint but rather what parts needs to be there. I can tell you 40% of what the system will look like five years from now, but the rest will be clarified along the way. It is not possible to blueprint the whole system (which would require lots of efforts!) and then build it. The whole software industry is going towards iterative methodologies.

In another project, there was a perceived need to improve the programs drastically. But there was no direct money involved, not until a customer said they would not buy our core products unless we provided a better tool for our analyses. Large money involved means that there is a motivation and a driving force, an immediate need for a project. In our integration discussions, the driving force is not there.

7 & 8.

As described, new development of data storage, which should be the integration point for the future. If we define it well and provide access methods in different languages, people can write their programs in the language that suits their needs best – FORTRAN, C++, Java, or whatever. Definition of a common data model has also been discussed, but there seems to be too big differences to complete before the development and iteration of the data storage.

The motivation is that one needs to focus on the immediate needs instead of visions many years into the future, and I am convinced the common data storage format enables data exchange to the extent needed.

The major concern with the Swedish data management system is that it too tightly coupled. Our current project will help breaking it apart, so that we can reuse different components. However, the integration point must be clear.

9 & 10.

We have tried spending long discussions defining a concrete vision, and we are also trying to do something useful on a shorter term that could (should) be the integration point for a looser integration. The current situation is that the use of the Swedish data management system here at the US site is very far away. The project of providing data storage is currently on hold, waiting for further requirements from the customers. But several years have passed, and we are not very much closer than when we started.

It would always be easier to ignore the Swedish data management system; it delays the schedule to always take relationship with other systems into account.

11.

Lessons learned:

- You will have lots of time-consuming discussions. This will require experienced people. It is easy to fall into philosophical discussions.
- You need to get buy-in. There must be strong financial support and commitment for the integration, otherwise the discussions will not lead to anything concrete, and integration will not be achieved.
- Do not try to define the whole system at once; it is too complicated to get the whole picture. Instead, find an integration point, and then focus on small deliveries with short intervals (3 months).
- You need to deliver; users must see immediate benefit of the deliveries.

12.

I try not to fall into philosophical discussions. They can be interesting, but you get lost. You must produce something.

In a complex system, not only one language or one technology will be used. This must be allowed. Only, the integration must be clear.

Interview I_{F1c}

1 & 2.

It is difficult to tell exactly what systems are part of the integration discussions. Here in Sweden, the core consists of the 3D simulator and the data management system, and in US it consists of the 3D simulator.

2.

The Swedish simulator in its current form has been in use 10-15 years, but has a heritage from the seventies. The Swedish data management system has been 7-8 years in production (development started in 1990). When that project started it had basically the same goals within the Swedish organization as we have today: to create a homogeneous simulation environment.

At the US site, they have their 3D simulator and some helper programs for some tasks that to some extent cover the functionality of the Swedish data management system.

A parallel change in the organization is that programming has previously not been an own discipline but has been carried out at the technical units, each with its own programming philosophy. Today we have centralized software development so that most of it is done in to one organizational unit at the Swedish site.

3.

A common calculation system for simulations of type A/B is believed to be of benefit for internal users and customers.

Also, a common system would mean saving on maintenance.

4.

All source code available.

5.

[Not directly discussed]

6.

We had meetings from early after the merger (late 2000).

For example, one person (user/manager) made a demonstration of the Swedish data management system at the US site, as well as I did myself at a US customer meeting. The US group also included this system in an evaluation done before embarking on a new development project a few years ago.

During fall 2002 we had meetings with groups representing users and technical expertise. Based on their evaluation of the existing systems, the concerned managers made a decision that felt very watered-down compared to what the user groups asked for.

After that not much happened. From the 3D simulators came the requirement on a new portable file storage format and a new (common) data model. It was very urgent for the US simulator. A project has been run, led by interviewee I_{F1b} to create a common data model, to be a part of the US & Swedish data management system as well as for stand-alone calculations.

There was a meeting early 2004 with involved managers, project leaders and some end-users. The vision outlined at that meeting is a common 3D simulator, a common data model and structure, and the Swedish data management system. This is the core of the simulation environment but there are also other related programs that will be connected as well.

We need to share the big vision centrally so that all development is carried out within the right projects. If possible each sub-project may be run entirely at one site for efficiency reasons, but must always be monitored by one common stakeholder group. Currently Projects run fairly freely and separately, and it is easy that we create sub-optimal solutions.

Meetings with different people in different roles throughout several years have contributed to a vision being outlined, little by little. My feeling is that we at the Swedish site have been more progressive and the vision has gained some commitment here. The future 3D simulator we have started developing will, when properly validated replace 3 simulators here.

In retrospect, as it has apparently been so difficult to agree, one can well ask whether it is worth the effort? Maybe we should run our businesses separately, the US site and the Swedish? Integration is not a goal in itself.

7.

(Interviewee not a technician.) Integration focused around common data format and data model, and the Swedish data management system.

8.

Why not a larger, totally integrated system? (I guess it is practically impossible from cost and time required?) Yes, even at the Swedish site this is something we after all these years still have not managed to do (even if we have made some progress lately). By the time such a system was ready, it would probably be obsolete. The best we can hope for is to find a way to create “seamless” (from the end-users perspective), well-defined interfaces between the various sub-systems, to make it possible to develop and maintain the sub-systems individually. Of course the interfaces and the functionality of the sub-systems will not stay the same for ever, as time goes by some functions will become obsolete, others will be added and some will be merged. It is also not only a question of cost and time, the number of stakeholders would also make it more difficult.

9.

The common data model project (run at the US site, with technical advisors from the Swedish site) has converged and diverged alternatively. The idea was that the project should address the common needs, but in practice it mainly focused on the US 3D simulator’s needs. The project so far has not resulted in any released product that meets our requirements and the formal QA status is questionable, basically due to lack of resources and proper management. The lead programmer of the project, interviewee I_{F1b}, also was the project manager. In retrospect we can see that this was probably not a good idea, the project turned out to be too complex for one person to handle both these functions. It has also been an ungrateful task to run this project, since the stakeholders have often been too busy with other projects to give good input to this project. A more experienced project manager should have been involved to take care of the project administration (including the formal QA) and communication with the stake holders.

A first attempt to design a common API for the 3D simulators was made during spring 2003.

We have started to create a common platform for further development in all programs: modules for managing physical units, for adapting to different national standards (language, physical units and more) and for error handling.

A recurring dilemma is lack of resources. Cooperation has suffered from this. There has been a lack of clear management and there has not been a strong vision broken down to “grass-root level”. Further, most software engineers are overloaded with work. With too little people and too little planning, it is of course difficult to do the things we want, and also to find the time to plan the future properly (together). The lack of resources has its causes outside of the integration efforts – the organization as a whole apparently (often for good reasons) prioritizes other things, such as customer delivery projects and other development projects that have already continued for a number of years.

The most concrete integration result is case F3. In this project as well as case F2, the US site has gained the most from our efforts. It must be a matter of give and take, but if it is always the same part that gives, something is wrong.

This is the latest but probably not the last chapter in our integration efforts. If someone is to blame for the current situation, it should be management – there is a clear discrepancy between the vision and the resource assignment.

But we can ask: what is the value of integration? From where comes the driving force? We spend lots of money to hopefully save money in the long run – but will we really save the same amount of money? What about usability – maybe a common type A/B calculation system is not worth that much for customers? Putting it to the edge: someone needs to decide whether we want integration or not. It is a question about value for the money. However, we can live without integration for the moment, but in the long run this might not be true.

10.

We do not have a final system yet.

11.

- If you are going to start something like this, you have to do it with commitment and resources. Make realistic plans and goals and assign appropriate resources.
- For relatively subordinate technology choices (in our case, how to represent and handle physical units) – just choose one so that we can make progress!
- A steering document, the level under the vision is needed, as a guide to projects. It is extra important in integration to put on paper what you agree about, and stick to it. (We have experienced that we agree when we meet but diverge as soon as we are back home and start working. Distributed development makes this even more important, as well as cultural differences.) More formalism than usual is needed. It must be allowed that these projects are more unwieldy than others to ensure that the different parties agree.
- A strong management that can commit to and enforce their decisions.
- You must meet often – we must meet in person more often. At a minimum every three months. Weekly meetings on the phone are not sufficient, more personal contact is needed. Even job rotation is a good idea, for someone to sit at the other site at least half a year or so. This will create understanding for each other’s culture.
- “Not invented here” is a strong factor, difficult to overcome. It requires a strong person at the right level to make the necessary decisions.

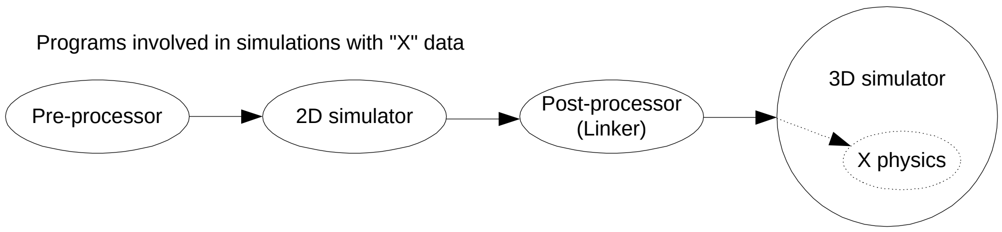
12.

I have tried to practice what I have said. Everyone realizes that we must meet more often, but each travel is a cost, but again; it is all a question of priorities and resources.

Interview I_{F2a}

1 & 2.

There is a certain kind of physics data, “X”, which is part of the simulations performed in this business. The basic architecture of the programs involved is as follows: there is a 2D simulator for X data, preparing input data to the 3D simulator which also contains an X data module. Before the 2D simulator there is a preprocessor which e.g. allows for setting up multiple cases to be run. There is also a postprocessor, which takes the output data and “links” it to the 3D simulator. The 3D simulator contains much more than the kind of data represented by the 2D simulator and pre-and postprocessor. The 3D simulator is “king”, in the sense being in the center of several activities (many other simulation or processing programs can be seen as preparing input to or processing output from the 3D simulator) as well as being a large and complex program.



The old 2D simulator was developed in Sweden long ago, and was given for free to the US site during the eighties during cooperation between the two by then different companies. The 2D simulator was developed in Sweden for the Swedish company’s need, which was for a certain kind of simulations, “A”. The US company modified it for their main needs, “B” simulations. So the 2D simulator was very similar (however, it contains basically only mathematical equations describing the physics involved, and the competitors’ corresponding simulators are also very similar). After that, both companies added pre- and-postprocessors on their own, preparing “A” or “B” data respectively, and linking to their respective 3D simulators.

3.

The companies merged in 2000. Some time during 2002 the US site began to develop a new 2D simulator. The main reason was that they wanted a fundamentally new simulation approach, where “all” cases would be simulated once and for all, instead of running cases per delivery project (in practice identical). At the same time, at the Swedish side a number of deficiencies in their model hade been experienced for a time, both in the 2D and 3D simulators. One example was that a greater flexibility was needed specifying which cases should be run (not “all” for all parameters, but perhaps “every second” for a certain physical dimension), and what data to store. As the two sites were by now part of the same company, this was identified as an integration opportunity. In the far future, the goal is to have a single, common 3D simulator (covering both simulations of type “A” and “B”) as well, so it was natural to cooperate and develop a common 2D simulator and pre-processor, as well as the model covering this particular physics inside the 3D simulator.

There were no particular goals for the integration, but both sides wanted to develop a next generation system, based on the deficiencies experienced with the current systems.

4.

All source code and documentation were owned by the company. The old 2D simulators were as said two variants of the same code, branched in the eighties for “A” and “B” calculations, now completely owned by the same company.

5.

Customers. The US site has many customers, and we are a little worried of what the consequences are when we have developed a completely new simulation chain. It is a major work to recalculate all data, perhaps from the last twelve years. It will be part of the project that our company performs this re-calculation on our own expense for the customers.

Other projects. Case F1, and a lot of "down-stream" programs will be affected by the new model – however, the 3D simulator is "king", and others have to adapt. In the US, the storage format will be completely new (today, all data is organized based on the assumption that the calculations are delivery project specific, which will be modified to the "once and for all" approach described above – for example, all input and output data are stored in the same file). There will be a number of migration solutions with translators, but only for programs and formats that will be phased out; the programs development is focused on will be modified so they use the new storage formats.

Users. Our (Swedish) users are tired of changes, although they of course want to be able to perform as accurate calculations as possible.

6.

No structured process was used. The basic architecture (sequence of preprocessor, 2D simulator, postprocessor, 3D simulator) was not questioned. Java was considered a modern language, discussed early during the integration discussions. It was considered especially well suited for GUIs (something envisioned in the future).

7 & 8.

The postprocessor is structured into two main layers and written in C++ and Tcl. This was because there was a belief that it would be possible to reuse parts from the previous Swedish postprocessor. It is a nice division between C++ and Tcl, where the basic data objects are implemented in C++ and the (large) physics equations are implemented in Tcl. In this program, every line of Tcl is equivalent of perhaps 20 lines of C++ code, so by using Tcl it becomes possible to gain overview of the equations (perhaps 20 lines of Tcl code). At this high level, there are no particular performance penalties by using a high-level scripting language such as Tcl. Personally, the interviewee would prefer Python (being object-oriented and more structured) to Tcl, but Tcl is widely used within the company.

The preprocessor was the first part being implemented, and the language chosen was Java, based on the early discussions on Java being a modern language. However, the interviewee thinks it was a bad decision. Many other programs are based on the C++/Tcl paradigm, and this will be the future for more and more programs (refer to case F1).

9.

Much work was put into the model before implementing. However, the "once and for all" approach was new to the US site. Also, the physical distance slowed down the progress.

The project ran into large problems. As the postprocessor was being implemented through reusing main parts of the previous Swedish postprocessor, it became obvious that it relied on fundamental assumptions that were no longer valid. One major example was the use of "dense" matrices (where all slots are filled with data), which would be exchanged to "sparse" matrices, allowing for saving storage space with a factor 20 to 50. During this phase, the interviewee was involved, and worked extremely hard together with another project member (Interviewee I_{F2d}). This problem was due to the fact that no proper analysis had been made, there were not enough staff available, and a lot of things were assumed in order to make progress.

By this time, new C++ classes had been implemented, so it was no practical option to change for Java (however, the interviewee is very glad for that afterwards). During the most difficult parts, they practiced pair-programming, and the interviewee is very positive to this. The two complemented each other very well, interviewee I_{F2d} being a physicist and the interviewee being the expert programmer. The model has itself also evolved during the work, but this should be seen as natural and was not the problem. The foundation of new model was well described, so that if the specification had been studied properly, reusing the old postprocessor would not have been attempted.

Another mistake made was that the new 2D simulator being developed in the US was forced to use the old output data format used by the old 2D simulator. This was because no proper analysis was made. It is messy, hard to read and error prone. It is not self-descriptive, but extra information is needed to be able to interpret the (binary) data. A work-around that has been practiced has been to print comments of a specified format with the output data. These comments is then used by any post-processing programs. The current standpoint is that when the new 2D simulator supports type "A" simulations, the file format will be exchanged.

10.

The final system 1.0 is planned to be released in February 2005 (one month after the interview). But a number of "snapshots" have been released, mainly in the US, and been evaluated for type "B" simulations. A number of problems have been found in both the model and the code, and changes have been performed.

11.

(Same answer as for case F1.) Most important is to sit together physically. Periodical visits, lasting one or two weeks, are required. For critical tasks, e.g. high-level design, longer periods of working together are needed, maybe one month. One cannot rely on tele- and video-conferencing.

12.

(Same answer as for case F1.) No, not really. Simultaneously and independently, new processes have been adopted from the US site (as a result of the merger). However, this integration project is a bad example of these, as no particular defined process was followed.

Interview I_{F2b}

1 & 2.

Previously there were two chains of preprocessor, 2D cross-section simulator, postprocessor and 3D simulator, one in the US and one in Sweden. In Sweden, these programs are written in FORTRAN, except the postprocessor which was written in C++/Tcl. The Swedish and US programs are aimed at two different main types of simulations, “A” and “B” respectively. In the future, there should be only one chain of these programs.

3.

The interviewee is not sure. When he was involved, there was already a decision to make a new preprocessor for the two 2D simulators.

4.

Everything was available.

5.

Users: No Swedish users have been involved, but in the US, an input data expert has been involved. The new systems describe the same physics, and will impose fewer restrictions, so it is hoped that the users will accept the new system.

Some physicists have been heavily involved; they are responsible for the models.

For a while, there was no project management, later there was a manager appointed (Per).

6.

There were two meetings, in (circa) November 2002 and February 2003, where the participants sketched what the new preprocessor should do. The interviewee is a software programmer, and much of his task was to understand the terminology and understand the existing systems.

After this phase, only the interviewee was assigned to the project, which was a very lonely task. There was no project management in place. The very long-term goals were clear (one single chain of programs), but no milestones and no time schedule was specified.

The requirements and design have evolved driven by the programming. Only when something is already implemented they suggest changes.

In US, they created their “B” model from the user manual for their existing preprocessor, which led to some peculiarities that were later changed.

The cooperation Sweden/US worked better than expected. Telephone conference once a week, and travels every now and then. The interviewee had heard it was difficult to cooperate with US. There has been no “we vs. them” mentality.

One major flaw in the project setup is to not have a common CVS. Files are sent via email, and have to be merged by hand. Very low grade to the project. This should have been in place in the beginning.

7.

There was a focus on what the preprocessor should produce, and it was realized that (part of) the US preprocessor must be reused for the US 2D simulator. So the new preprocessor had to be given somewhat different tasks depending on which 2D simulator would be used in the next step.

The preprocessor is completely new, written in Java and based on XML. It is based on a new approach allowing more flexible input data. Previously, the input data format was very “stiff”, sometimes data was repeated, and the user had to support approximately 100 lines of input data in an ugly syntax; it was easy to make errors. Now, the input is more flexible (it is possible to change equations dynamically), easier to understand, is based on templates, and approximately 20 lines are needed.

The 2D simulators have been slightly adapted.

I think a new common module has been created for the 3D simulators.

Both the preprocessor and the postprocessor have become complex due to the fact that two 2D simulators are supported. The plan is to retire the Swedish 2D simulator and implement also the "B" type of simulations in the US simulator, and the best would perhaps had been to do this immediately. Unsure why the choice has been made to use both in parallel for a while.

8.

Java was a company policy. Python or Ruby would have been preferable, it would have been ca 10 times faster.

9.

We should have retired the Swedish 2D simulator early.

The model became more complex than initially believed. It was thought that it would be a simple matter of substitution, but more calculations were needed. Also, the preprocessor needs to forward certain data to the postprocessor, which complicates things. This will hopefully be avoided when the Swedish 2D simulator is replaced with the US simulator.

10.

The new system is not yet released. The pre- and post-processors are "almost" finished. The users will like the new input format, but the preprocessor is much slower than the old Swedish preprocessor. The advantages gained through Java are several however: easier for the programmer.

11.

Success factors:

- This is a special application, requiring specialist knowledge. As a programmer, you need a technical expert, assigned to the project with the role of being a "mentor". Lot of time was spent finding someone to ask, and there was always a feeling that you disturbed this other person, who had other things to do. In the US, there was a skilled user involved, preparing input data and running the cases, but in Sweden there was no such person. Now, however, there is one person assigned as a "mentor".

- You need to listen to those who are more doubtful. One person (who later became the interviewees mentor) said early: "Don't be discouraged, but I think this project will fail. It is easy to write a new preprocessor – if it was easy we would have replaced the old one long ago."
- A common development environment (some have programmed in Windows, others in UNIX, no common CVS).
- Project management is needed.
- Sitting together. When we visit each other much gets done, while on our own it feels heavy. We meet at least every second month.
- Involve a user.

Mistakes:

- A lot of assumptions were made early; I don’t know where they come from. No one verified them. It seemed to be undocumented information that everyone took for granted. More must be written down on paper! Input and output data formats need to be carefully specified.
- From outside, it appears as the project has had no problems, because the requirements specification has not formally changed – but that is because it has not yet been formally issued. If it had been issued one year ago, when it was first completed, it should now be in its fifth revision.

12.

We try to meet more often than in the beginning.
I now have a “mentor” (a technical expert in physics).

Interview I$_{F2c}$

1 & 2.

Not discussed. See other interviews.

The interviewee was involved approximately one and a half year ago, one or one and a half year after project start.

3.

There was a vision of having a common simulation system, including: the same preprocessor, 2D simulator and postprocessor as well as a common module within the two 3D simulators. The final vision also includes a single 3D simulator. There was a project document from one and a half years ago, when I was involved as project manager for the Swedish site. The following project motivation and main goals were identified:

- Integration of software development
- Use of common methods/codes for production calculations
- To improve the current level of productivity
- To decrease future software support and maintenance costs
- Improved methods accuracy and quality

Together with this, there were some high-level ideas on how it could be achieved, e.g. to adapt the existing Swedish postprocessor. The focus in this document was on the preprocessor, for which the ideas were perhaps too detailed, while for the postprocessor there was very rudimentary information. Later, it was the postprocessor that became the big problem.

4.

Everything available.

5.

- Higher management, who drives/orders/sponsors the project. It is financed both with development money and integration money (assigned within the company after the company merger).
- A person assigned as central expert on “A” type simulation methods.
- Managers for the related units (software development and model development) in both Sweden and the US.
- The end users, primarily the US users (doing type “B” simulations).
- A related company is being informed.

There are always conflicts between new development and maintenance of existing programs, when resources are limited.

6.

There were initial ideas and discussions that these simulations should be integrated, documented in a project plan. There were differences between the two main types of simulations being made within the company (type “A” mainly in Sweden, and type “B” in the US).

The driving force has been the planned delivery of a new version of the US 3D simulator which would include a new model. This is the first application of the new model and the new system. This has mostly been good, a good driving force. In the US, testers have been involved in parallel with development, and have contributed much (in Sweden, we have also had one tester). However, this means that it is the US needs have had the highest priority, and we have sometimes felt as being resources for their needs.

7 & 8.

The postprocessor: there was a three-layered architecture in the old Swedish postprocessor, with an expert software engineer (interviewee I_{F2a}) in Sweden responsible for the bottom layer in C++, a physics expert (interviewee I_{F2d}) in Sweden for the middle layer, also in C++, and a programmer from the US responsible for the top layer written in Tcl. The Tcl programming started assuming that the C++ layer could provide the necessary functionality with the existing C++ class definitions and interface routines. This design assumption, however, was found to be not valid due to changes in the theoretical data structure to be modeled by the code. Unfortunately, this fact was not realized until about a year ago. The conclusion was that all the Tcl work performed that far was wasted.

9.

The new "once-through-data" methodology to be used has made a larger impact on the US programs than was initially understood. The requirements evolved in parallel with the development, and to some extent begun to involve not only the data generation methodology for the 3D simulator, but also the downstream programs after the 3D simulator. I said "stop" half a year ago, and said that the task of this project is to develop code that can generate data to the 3D simulator according to the new methodology and to assure that 3D simulator is modified to handle these new data.

It has been problematic to keep the requirements stable. New opinions and improvements have vitalized the project, which is good, but it has been difficult to administer. Currently, we have teleconference once a week, where we keep an action list updated; it is in a very simple format: a bulleted list of things to be made, with different colors and fonts denoting priorities and what has been done.

There was a point in time when it was decided that the old Swedish postprocessor was not sufficient, and had to be mostly rewritten. This incurred a project delay of at least half a year.

The project has been rescheduled several times. Nothing of the original project plan has been correct. The budget has been assigned per year, and we have been almost within budget each year. But it takes longer than expected, which of course makes the final cost higher than initially planned for.

Organizational things: people have been brought together. Weekly teleconferences and week-long visits in either direction approximately every third month.

10.

The system has been tested mostly in the US, where people are "satisfied" in some sense. For type "A" simulations we actually do not know yet if it will work. We have seen some specific fundamental problems, for example with lost precision in some cases. To solve this type of questions is rather to be considered as research than development work, since you don't know if you may reach the objective of the whole development effort.

The preprocessor is in one sense a single program, but it contains large parts that are "A" specific or "B" specific. Also the old file format has been kept, but the data defined is to a large extent different for the two types of simulations. Maybe there are so fundamental differences in the physics that it is impossible to do a better integration?

The project members have learned a lot, as well as the company as a whole. People have become acquainted. There is less of a "we and them" feeling and more of "we together".

11.

- Do a proper pre-study; analyze everything that may impact the project.
- You need to have an overall vision of the goal, what you want to achieve. But you have to limit your scope, be systematic and perform one step at a time. Do not try to do too much at once. Do this integration with one limited application at a time; implement it to be able to see how fundamental and serious the differences are.

- Involve people from both organizations. Job rotation (to sit at the other site during periods). This will generate a feeling for a common goal.
- The need for sitting together is particularly emphasized during design. The requirements are at first at such a high level so that it is easy to agree. When it comes to design (or detailed requirements) you have to consider the detailed differences. In this project, the (detailed) model was the big difficulty to integrate, not agreeing on requirements like “shall use this or that file format”.
- Involving the end users – how can this be done? It is always very difficult. The US counterpart has known the need. Here in Sweden, for “A” type simulations, there has (as said) not been a driving force from the end users, but in the future they will have to use this system nevertheless.

12.

- I have become more “realistic” for this type of “research” development projects. I have a deeper appreciation of the problems. I am not the “police” type. But in other type of projects I manage, such as specific delivery projects to external customers, I act more strictly and harder.
- The last two years, there has been a higher pressure within the organization. Previously, there was more time to do things good and right.
- A good thing with a company merger is that you can work over the borders; there is a feeling of fellowship

Interview I_{F2d}

1 & 2.

See other interviews. Nothing new. This discussion mostly concerns the development of the postprocessor.

3.

Reasons:

- To save resource for software development and maintenance.
- In the US, there was an interest for “once and for all” simulations.
- There was a potential to reuse the module in 3D simulator in another type of 3D simulator.
- User friendliness – a great improvement.
- A larger base of users, who can more easily do new types of jobs (type “A” and “B” simulations)
- A larger base of testers (users)

Sometimes it is maybe not worth to integrate? Is the product modern enough? Some skepticism towards the new 2D simulator being developed. Our competitors already have what we are developing.

4.

Everything available.

5.

A number of method developers/model developers were involved early, contributing expert physics knowledge.

Software designer – an expert software engineer was involved to help the project through.

Users – a few testers. But the part discussed is internal.

Managers – someone must have initiated the project, and assigned resources, although it was not properly managed.

6.

Some time after the company merger, some meetings were held discussing the future of this kind of simulations. Some benchmark calculations were made with both the Swedish and US systems (chains of preprocessor, 2D simulator, postprocessor, 3D simulator). After a while, someone (in the US) said that we will have only one system for this particular kind of simulations, including a delivery date.

There was no clear project start; suddenly people were responsible for different things. No proper analysis was done, it was decided/assumed that the old Swedish 2D postprocessor would be the starting point. This decision was made based on engineering judgments; we thought that this was the best choice at this time.

There were large problems with agreeing on the requirements, and the effort needed was grossly underestimated. The requirements have grown almost by themselves. Like water filling the available empty space. However, this was quite expected. Also, I was perhaps not enough clear-sighted to handle all the requirement changes in the best way.

7 & 8.

- The old file format (between old Swedish 2D simulator and postprocessor) has been kept, and it was decided that the new US 2D simulator had to use the same (instead of using some more modern and more structured format). This was because it was initially believed that the old Swedish postprocessor would be reused to a high degree, which was not the case. Now, the only part that remains is the interface to the old file format! It was a good thought, but the fundamental problem was that the model was so different.

- Two different ways of reading data. (Most of it in common, but also a number of differences between the two 2D simulators.)
- The (mathematical) model is totally different, based on a different approach. Of course, the bottom of the physics is the same – the system serves the same purpose as before, is used to analyze the same physical reality.
- The modules of the 3D simulators are also totally new, but there are similarities with the old modules.
- The layered structure, the concept of dividing the program into C++/Tcl remains. This paradigm was used both in the old Swedish postprocessor as well as in other places within the company.

It is a good thing we have not reused so much. In the case where we did reuse something (the old file format) it was a mistake. This file format will be exchanged in the future, when the old Swedish 2D simulator is retired and the newer US 2D simulator takes over "A" type simulations.

9.

The delivery will be in 1 month, one and a half year late. Among the problems are: many people involved, the Atlantic in between. Lack of resources (= other projects could disturb the focus on this system.)

The initial assumptions did not hold. In retrospect, it would maybe have been wiser to retire the Swedish 2D simulator at once, instead of adapting other programs to it and its old file format.

A software expert (interviewee I_{F2a}) and the interviewee worked very close together during a period. We did Extreme Programming in our own way, e.g. by sitting together, software expert and the physics specialist and model developer. Also, I have been careful to keep code and documentation up to date. Different people need different information, and it has to be consistent: the code, design descriptions, the model (physics equations etc.), and the interface descriptions.

10.

Difficult to answer – the expectations were increased during the project, as the requirements grew. It is a technical success though, both from a physics perspective and considering the software itself. The new postprocessor is many times better than the old postprocessor. It is well structured; we use C++ in a proper way, e.g. using iterators instead of pointers. Although the decision to keep the old file format is a mistake, this will be corrected in the future. The core is very good and can be considered a technical success: we succeeded in integrating two worlds – although with much pain.

11.

- Having the right competence. At first we did not get the right people, but involving the software expert became a success.
- Planning. You cannot promise a delivery date before you know what to deliver.
- Requirements must be clear. Proper analysis must be done. (This relates to having the right people.)
- Feedback. Sometimes I had to wait for feedback. Partly this is due to the different time zones. Many times, you had to communicate via email instead of having a short meeting.

12.

We involved the right people. After a while, there was proper project management.

We have had regular meetings where we have discussed status. Teleconference. Communication becomes much easier when you agree, when you have more specific questions.

Interview I_{F2e}

1 & 2.

The Swedish 3D simulator, for type “A” calculations, was first released during the seventies, with a major update around 1990. The US simulator, for type “B” calculations, was first released in the middle of the eighties. Both contain several hundred KLOC. There was also a third system in use at one US site.

In both simulators there are several main modules for different aspects of the physics involved. The mathematics involved is very complex: differential equations, non-linear solutions (all parts/modules affect all others); lots of matrix operations, inversions and eigenvalue problems. In addition to these modules, there are modules for post-processing, for input and output handling, for different types of automation tasks.

These simulators are used to support certain processes in the company’s business. After the company merger in 2000, there were discussions how to be able to use the same programs throughout the company in order to make maintenance easier. Initially, several important integration projects were discussed.

3.

Motivations:

- The same system should be used for both type A and type B simulations.
- Decrease maintenance costs.
- Smaller number of procedures used within the company for the same main type of business.
- Make usage simpler for users. This will decrease the number of errors made in the process – today you almost have to be an expert in how to use system X (a related system).
- Improve accuracy and robustness in the mathematical models.
- Apply the same model both for dynamic and static calculations.

4.

Some open source libraries have been used for standard mathematical procedures. Otherwise, the systems as such are completely owned and controlled by the company.

5.

Model developers.

There was one model development group from the US, and one from Sweden, each consisting of about four people. Also one or two people from the other US site, people who at least knew something about their software systems. Also some people from cooperating companies were involved and had opinions.

Users.

The user involvement has not been a dominant part in the process. But users have previously complained about the complex interface. There have been complex input file formats, and the Swedish system has not given proper error messages early in the chain (at the pre-processor stage). The US system included more sophisticated automation tools early in the chain (ALPHA) already since the eighties, so the US developers brought much knowledge how it should be done. Our Swedish users have seen other systems and know how it can be done better. We have had users review (informally?) the system during the development.

Managers.

There was a lot of optimism in the beginning, but managers and project leaders have grossly underestimated the difficulties (see further discussion under question 9 below).

6.

Initially, the model developers (including me) studied the other system's documentation of the mathematical models, considered similarities and differences, and wrote down ideas of future models. This also included an early effort estimation (without considering the design of the actual software).

USA was on its way to develop something new, and their ideas were similar to what we had done. We had some problems with our models. Our response to the US was "we know of a few features you should have, and of some drawbacks as well". So there was a win-win situation, and both parties already had plans or ideas for further major development: they (US) wanted to do something new anyway, and we wanted to improve our models (especially for transient analyses).

7.

The preprocessor was completely new. In Sweden, we had nothing similar, but in the US they had a pre-processor with some of the desired functionality.

The post-processor was completely new, but started from our Swedish postprocessor. In the end, only a few parts were reused.

A new file format was defined for the data transferred from the post-processor to the 3D simulator.

The model inside the 3D simulator was new, including a reconstruction module reading the output file from the post-processor. First, an approach was tried where a temporary data interface was defined to facilitate communication between the new model and the two old simulators. This meant a lot of translations between formats and often coding difficulties were encountered in communicating between new and old code. Also, error handling was not standardized and each developer adopted whatever system was available at their site. This lead to incompatible coding that eventually put the developers into positions where unwillingly neglected proper error handling. Quality of the product was at risk. This lead to the approach that we switched to later, to define a common data structure, as well as common error handling routines. It required a week's meeting to agree, and then the new base was implemented in about two months. This is the future vision, so all new development will be based on this data structure and use of this error handling package. In this way, all future merges/integrations will be smoother. You need to do the fundamentals first; it is dangerous to think "I will fix this later". Sometimes the management push too hard to see results so that working on the fundaments is not allowed.

8.

Basically everything needed to be rewritten, because of the following reasons:

- The two existing systems (the decision had been taken that the system used at the third site would not be a candidate for integration since at least one of the existing systems could be modified to satisfy that site's requirements) represented two somewhat different approaches to compute the same thing. It was decided that the Swedish approach, which was somewhat more generic, should be kept.
- The Swedish module was badly implemented – spaghetti. It has been very hard to maintain, and it was decided at the start that it was in such a bad shape so that it was not possible to reuse. It was written in FORTRAN 77, and only a few things had been updated to reflect more recent good practices.

In the Swedish 3D simulator, there will be a possibility to choose between using the old and the new model for some years into the future. This will enable a period when the new model can relatively easily be validated against the old model in case of seemingly strange results. Also, some customers may simply not want to use the new model.

9.

Managers and project leaders have grossly underestimated the difficulties. They did not realize that you need the right people. In this project, the wrong people were assigned, which wasted a year. The Swedish site got the most of the work, since it was too difficult to release the desired US people from their current tasks. To be fair, it should be said that all involved has learned to “lie” and give underestimations of the efforts required. In my experience, physics model software projects will take 4-5 times longer than the estimates given by even the most experienced developer (this is in calendar time, not necessarily man-time).

In this project, the amount of rewrite was grossly underestimated. The existing code communicated through FORTRAN Common blocks, yielding complex control dependencies. It is not possible to just take existing routines and integrate. The existing code is written in an old fashion. It is not possible to take the part that is best from each of the systems and put together into a new system, because there are so many dependencies. It has not been at all as well modularized as can be done with e.g. object oriented programming. It is better today with FORTRAN 95, which allows encapsulation. At a high level, managers deceived themselves by saying “let’s take different parts and put them together”.

Two main factors in this project taking so much longer than planned:

- The wrong resources/people were assigned.
- There was a gross underestimation from the part of the managers – they did not want to understand the complexity of the problem. Sometimes they had a brief talk with a technical person and asked “will this work?”, got the answer “yes”, and returned to higher management with these assertions.

10.

I consider it a success for the US system (which is currently being released with the new model). In the Swedish system we will also see many advantages when the new model is included. However, nothing is verified for the Swedish system yet.

11.

Fundamental:

- The right people must be assigned to the right tasks. It is not possible to train people for these kinds of tasks when the project already has deadlines set.
- Good communication is essential; you have to have meetings often.
- A common development platform, such as the same CVS system, same compilers, same operating system. We still do not have this, and always drift apart after meetings. We have to have new meetings to synchronize our efforts. This should not be necessary.

12.

As described, we now have a common data structure and common error handling, which will make integration of future modules in the simulator much easier.

On the other hand, we do not have a common development environment (CVS, compilers, etc.) and will continue to run into the same problems due to that.

Interview I_{F2f}

1 & 2.

The first version of the Swedish 3D simulator was created during the sixties, for type “A” simulations. Throughout the years, it has been extended and maintained well. Under the surface, some structural changes have been made. In the middle of the eighties, a version for type “B” simulations was created, and in the nineties they merged to an A/B version of the simulator. It now consists of ~250 KLOC of FORTRAN code.

The structure can be described as a main program invoking different modules for different kinds of physics. These modules are clearly visible in the source code structure, but have lots of dependencies, e.g. through sharing of data. It does not have the clean modularization it could have using the “module” construct of FORTRAN 90 and 95. Little by little, the code has been restructured towards this, but only a small part has been renewed. This has not caused particularly many bugs, but makes it difficult to maintain and update the code. It is difficult to understand the consequences of a change, things are scattered. Maintaining the system would require a decade of learning – any large, complex system would require a long learning curve, no matter how well structured. But you sleep better at night if you perform changes in a well designed system, you are more sure you did something correct.

I do not know the US system well, but I believe it is written in FORTRAN and is fairly aged.

3.

There is a vision of having one common 3D simulator. The advantages would be to only need to maintain one system, the two systems to a large extent overlaps functionally.

4.

Everything owned and controlled in-house.

5.

Maintainers/system responsible: I and interviewee I_{F2e} have been most influential from the Swedish side.

Users: Users have had little influence. The indirect influence is the requirements. The overall requirement is that the future 3D simulator must be able to do everything the two simulators can do today, as well as the new functionality being on the relative immediate development agenda. To support future (unknown) extensions, the system must make less assumptions, should support data that might be needed in the future (with some reasonable balance of course).

External customers: In practice, a long period of supporting the existing systems will be required. It is not enough to make 90% of the customers (and internal users) change to a new version or generation of a system; you will still have to support the old system for a long time.

Managers: It is difficult to say how much managers have been driving in this process. The slow process so far probably is due to a lack of time, solving daily problems at each site probably takes lots of the managers’ efforts. There is always a new release around the corner and little time for strategic tasks (like e.g. updating the structure, or working towards an integration).

6.

I have only been involved a few times. The future common 3D simulator is still in its cradle.

Less than a year ago, I and interviewee I_{F2e} met the Americans to outline a future simulator. This was the first time I met the American people. It was a major step, just to get to know who are involved. The high-level question marks, like “why do they do it in this way” are clarified when you discuss personally. Most things that seemed strange have natural explanations. Interviewee I_{F2e} and some others had sketched before, but we were more people this time who refined and finalized the decisions. During a couple of days we went through each parameter of the data model, discussed possible changes etc.

We agreed that we should not be hindered by today's systems but choose a design as we would like to have it in the future. Nevertheless, sometimes we argued that "we should do it this way because this the way we do it today". You are always colored by the background of your own system (maybe someone else would have come up with a different solution) but in the end we think we defined a good data model for the future. It is not yet implemented in the Swedish system; it will be shown later how well it works in reality. It would be good if we had already started implementing it, partly for getting experience and evaluating the model, and partly to not fall behind. Since the Swedish system can do both type A and B simulations, it would not be good for the overall result if the US system (which can only do type B calculations) will be much more advanced from this perspective.

We did not really discuss the cost of implementation. A new 3D simulator would take ~10 years until fully validated and ready for use. Even if we would do as little as possible to merge the simulators, still a number of years would be required. With this in mind, the vision must be to create something that is better than the existing systems, not just a minimal merge.

A vision of where we want to reach is needed, but also some more concrete steps along the way. A vision without breaking down the way there into steps will be too abstract, and the two systems are improved stepwise without a vision, they will likely never merge into one. Clearly there is an additional cost in keeping the systems alive while working towards the vision, compared to focusing all efforts only on integration. However, it would not necessarily cost very much more to aim towards the vision during each step. Breaking down the process into steps is also important since you learn along the way. In the beginning you do not know as much as you do in the end. You can not do everything at once.

7.

To arrive at a common simulator, a common data model has been defined, and interfaces at the level of functions and argument lists. These are well specified and documented, and the functions must have no unknown effects or extra sources of data, e.g. reading from files.

I remember once (earlier), when someone asked me to send the module for a certain kind of physics simulations, probably to test it for use somewhere else. It was very difficult to isolate it from the program, due to all kinds of dependencies (see question 1 & 2). Managers seem overly optimistic when believing that it is simple to just take one functional module out of the system and reuse it. It is a complex system, and it is not designed with reuse of separate modules in mind.

We have not yet implemented any of this in the Swedish system. It will be apparent as time passes whether this is a suitable approach.

8.

To enable integration of physical models in a "clean" structure. We want to create not only a "minimal" integration with only today's functionality, but rather a new simulator generation which should be easy to maintain.

It is not the case that some parts of one system are so much better than in the other. The models are somewhat different, but are both advanced enough and not totally dissimilar. The slow changes are more due to different file formats and everything that implies (support for existing data, users having to learn a new system). Synchronizing the outside, in terms of the interface, is much more complex than the inside, since you need to have approval from the users.

9 & 10.

We are still in the beginning of the process, but “so far so good”. I am a little concerned about the time spent so far, though – if we did this little during the past two years, how much longer will it take to reach the vision? I hope the integration efforts will be intensified at some time. Possibly, a certain momentum might be gained once we get started, but for all time estimates it is difficult to say whether it is a wish, a hope, or a belief. The reason for the slow progress seems to be lack of time “for the moment” – but is it realistic to believe we will have more time to spare in five years from now?

11.

Important:

- To get to know each other, to meet, and to work together with something concrete. Sporadic contacts via telephone and email is not enough. You need to sit together during some significant amount of time. We have some experience from being located at two sites in the same city, 2 km apart, and the same effects are discernible.
- You must have a driving force in the project, an active project leader who is pushing. It is a large system, many people, several sites and several continents involved. This type of project will not run itself.

12.

We have discussed the need of a common source code repository and other technical things that might be barriers.

Interview I_{F3}

1 & 2.

Background: company merger in late 2000, between a company with a main US site (labeled US1 throughout the interview), and a company with a Swedish and a US site (labeled SE and US2). Software supports the core business.

Each of the three sites used software issue reporting systems, i.e. systems for reporting software bugs, software enhancement request and similar (in IEEE 1044.1-1995 [11] called “anomalies”). Not only reporting, but also implementing a workflow for resolving the issues, with features as e.g. notifications to different users. Two of them (SE and US2) were developed in-house in Lotus Notes, and the third (at US1) was the commercial product Rational ClearDDTS (installed in the early nineties, today there is a successor named IBM ClearQuest). The US2 system was the system with the best process support, and the SE system had the closest connection between the issue reporting system and the development process, for example by generating a large portion of release documentation from the issue database (resolved and known, not yet resolved issues). These existing systems are described to some extent in the degree project report D_{F3a} [25]. The two in-house developed systems were surprisingly similar, perhaps as a result of the way you have to develop in Lotus Notes.

There was also a common reporting system for any issue within the company, from sexual harassments to hardware manufacturing errors. One of the few exceptions from this rule was software issue reporting, which was acknowledged to have particular needs and was allowed to have specific issue handling mechanisms. None of the existing software issue reporting systems had any connection to this system.

3.

After the company merger, there was a desire to have only one system throughout the enterprise. There were two main motivations for this:

- Processes. For example, the phases of the workflows were different, the notification mechanisms were slightly different, and the attributes of the report were different (e.g. whether some attributes exist at all, and what different attributes mean). This meant that:
  - No common understanding between the sites of how software issues shall be handled - slow, resource consuming processes between the sites (as would be the case when exchanging data between any two different systems, say Word and Framemaker).
  - The company did not have a common interface towards customers. This includes both 1) a process interface, which types of notifications are sent and when, including terms used, and 2) a system interface, meaning a means for customers to access the system e.g. via the Internet. At the site using ClearDDTS, a proprietary system interface had been built, but the other sites had no similar interface for customers.
  - This will result in lower quality.
- Maintenance. Having three organizations maintaining and supporting three similar systems appear to be a waste of resources.
  - No central administration and maintenance of the existing systems and its data
  - Systems are old and sometimes hard to use – errors and enhancements not always reported, high administration
  - No connection to common issue reporting system – administration overhead

The three sites had slightly different attitudes:

- The US1 site, which had a ten-year old commercial system: “We would like to upgrade to a more modern system, but it must support our existing process.”
- The US2 site, which had the most advanced system from a QA perspective: “We see the value in changing to a common system, but it must be as similar to ours as possible.”

- The SE site, which had a close connection between the issue reporting system and the development process: “We want to tie the issue reporting system even closer to the development process.”

Also, there was a desire for interaction with the general issue reporting system within the company. Some issues should be in both systems, and there was (and is) a vision that the users shall not need to cut and paste information from one system into another, but that there should be some seamless, automated way of transferring the relevant information.

4.

The source code for the two Lotus Notes-based systems was completely owned in-house. Rational ClearDDTS was a commercial product (with an in-house developed extension).

5.

The stakeholders were (see D_{F3a} [25], p14):

- Manager at the highest level, who assigned the money and responsibility to the SE lower-level manager
- The lower-level managers of software development at the three sites
- Software developers, QA people, and users
- External customers and users, were consciously not involved

There were no real conflicts between stakeholders as such. If anything, there were differences in opinion between the sites concerning both the scope and implementation of the new system.

6.

There was a meeting with representatives for all three sites where higher management gave the SE site manager for the software development department the mission to introduce one single software issue reporting system. The participants agreed at a high level, and also wrote a document with business requirements. After this (August 2003) the SE site engaged two M.Sc. students for their degree project (one of which is the interviewee) – which shows that the project did not have very high priority. The SE site was the driving force, and the vision was higher than “only” issue reporting: an “information portal” (see title of D_{F3a}, the degree project report: “Design of an Information Portal for Engineers” [25]). In this work, detailed requirements on issue reporting functionality were collected from all three sites (this included a trip for the M.Sc. students to the US sites).

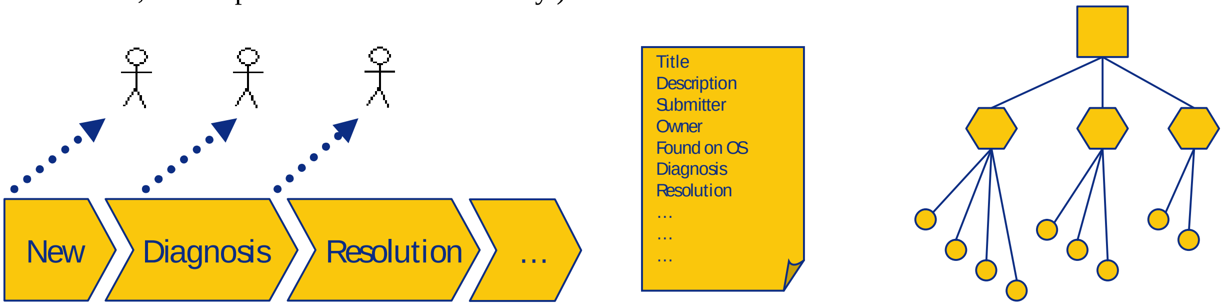
The US2 site wanted to keep the Lotus Notes in-house developed system. The US1 site wanted to find a commercial product on the market (they focused on IBM ClearQuest, the successor of their ClearDDTS system, with which they were familiar while it represented 10 years of improvements). The SE site manager’s focus was vendor independency, giving two main alternatives: building the system in-house or finding an open source solution. As the managers had different opinions, the M.Sc. students investigated the three main options: open source, building in-house, or finding a commercial product. This resulted in extensive documentation with the only purpose of decision-making. The M.Sc. design work (of the broader scope, an “information portal”) continued and was finished in January 2004, and when reviewed, it was considered that the work would be too large to implement internally, so the focus then remained on implementing a common issue reporting system.

After the degree project, it seemed obvious that an issue reporting system would be too large to build in-house, so the focus shifted towards the marketplace (commercial as well as open source). However, the project had low priority for a while, and there was a feeling that the project was back on scratch. You always learn something from this type of projects, however, so it would be wrong to say the efforts were wasted. There was another meeting where the old documents were studied again, and there was a determination that “we have to make this happen”. One of the M.Sc. students (the interviewee) stayed with the company and led this project. The inputs were the high-level business requirements and the detailed requirements already noted, and the task was to investigate if there were any suitable existing systems.

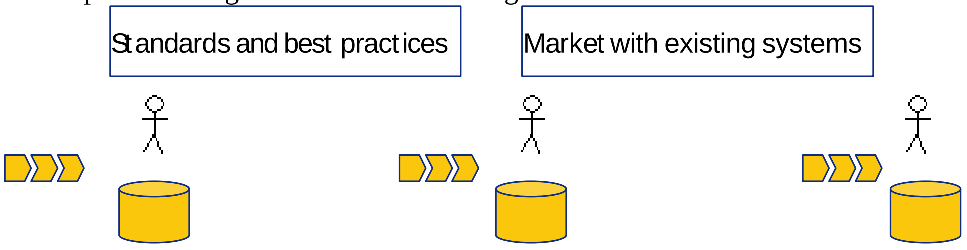
The requirements were rewritten as a feature list, to be able to evaluate requirements fulfillment for each system found. One goal was to unify the processes. Common process and business model wanted, and five particular goals were identified:

- Issue data model (attributes)
- Issue reporting and resolution process (workflow)
- User roles (Product Owners, Developers, End Users…)
- Notifications (internal and external)
- Product model (Product Families, Products, Versions)

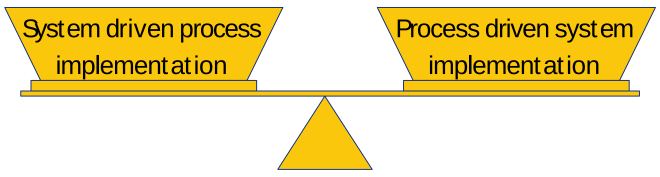
The figure below illustrates all these five points: to the left are users, a process, and notifications; in the middle is an issue, and to the right is a product model. (This and all of the following figures are provided by the interviewee, from a presentation held internally.)



There were three existing processes, systems, and user groups, cultures (the attitude towards issues and issue reporting and resolution). Where to start? And in addition to the three existing systems used internally, there are known standards (in particular IEEE 1044.1-1995) and best practices, as well as a commercial (and open source) marketplace. See figure below. How to merge these five sources?



One important question to ask is how mature the existing processes and data models within the company are – if standards, best practices and the market provide alternatives are more mature, they would improve the current state at the company. Depending on the current status in the company, the balance between two main strategies had to be found: System driven process implementation and Process driven system implementation. See figure below. Two main attitudes can be formulated: “we found a good tool, we have to adjust”, or “there is no appropriate tool available, we have to build one”. (A parallel: in US when buying a Superman costume in a toy store, there is a warning label: “Caution: Cape does not enable user to fly”. The tool is not the process! Learn how to fly first – then put on a cape to ease the ride!)


System driven process implementation
Process driven system implementation

We have in steps:

- Studied each others solutions internally at the three sites
- Identified best practices
- Looked at standards and tools
- Merged and enhanced!

In this process, all involved tried to avoid the trap of assuming “our own” system to be the best, but acknowledging the benefits of the others.

The outcome of the process: the existing processes followed the basics of the IEEE standard, considered mature. Next question: Build vs. Buy? How unique is our common process? Can an existing system be adapted to it? Cost of existing system vs. custom development?

A swift market survey showed that issue reporting systems is a mature market with many high-quality, customizable systems. 95 systems were initially found (by searching the Internet). Therefore, system driven process implementation was not forced upon us, and there was a belief that by customizing an existing product we could preserve the best of our existing processes. To evaluate the existing systems, a four-step iterative screening procedure was used, where in each step the systems to evaluate were given more evaluation time. Three criteria were used:

- Requirements fulfillment (feature list)
- “Ilities” (Scalability, Maintainability, Usability etc …)
- Vendor reliability

(The following description of the screening methodology used is taken directly from a document the interviewee provided and presented during the interview.)

(90 systems left) **Step 1**

Given all found systems, a number of systems will initially be rejected.

- Rejected system is outdated or unsupported
- Rejected system has obvious low requirements fulfillment (Requirements are given in document)

Evaluation time per system is max 5 minutes. (90 systems gives approx 6-7 mh total effort)

(20-25 systems left) **Step 2**

Given the accepted systems from 1, a number of systems will be rejected.

- Rejected system has low requirements fulfillment
- Rejected system has low customization/programmability possebilities

Additional time per system is 10 minutes. (20-25 systems gives approx 4-5 mh total effort)

(10 systems left) **Step 3**

Given the accepted systems from 2, 5 systems will be actively chosen.

- Chosen systems has advantages to other systems, motivations given.

Additional time per system is 20 minutes. (10 systems gives approx 3-4 mh total effort)

(5 systems left) **Step 4** A,B,C,D

Given the choosen systems from 3, one candidate will be chosen.

- The candidate system has the best detailed evaluation score and summary of the systems (see evaluation description below)

(one system left) **Step 5**

The candidate will be thuroughly tried out, all features tested and evaluated.

If the systems is found to have flaws, step 4 will be repeated with a new choice.

Systems evaluation in step 4

A. Requirements fulfillment.

- Graded feature list based on business requirements and their priorities
- Text comments in addition to grades, motivations

B. Statement on each tool regarding:

- Usability
- Maintainability
- Scalability
- Customizability (how much the system can be changed without touching the code)
- Programmability (ability to change/add features, CAPs connection, external interface)
- Ease of transfer of legacy data

C. License/Acquisition cost information,
based on business requirement

D. Information about Vendor – dependency, support, ability to add features etc.

Result:

Each system will get a score (based on A), a summary (based on A-D) and a final judgment. The conclusion of the evaluation should be a recommendation of 1 system, and the rest of the systems given in order of their suitability.

In phase 4, the requirements were listed for each of the 6 systems (one more than the desired number), and each feature graded (0-3) and assigned a weight (1-3) depending on importance. Five of the systems achieved 82-89% of requirements fulfillment (and the sixth 73%). Also, the “ilities” were evaluated and informally compared, with the result that all scored high. For vendor reliability, one of the systems was discarded as coming from a very small company with an impression of low experience in business (sales, support, maintenance, and training). Being so difficult to distinguish between the other five, the price became decisive. One of the companies gave a very good offer.

7.

A customizable commercial product was acquired and customized. As described above, the existing processes were merged and implemented.

8.

It would be a too major task, outside of the company's core business to create such a system in-house. However, the major two drawbacks are:

- Vendor dependency
- Vendor dependency (yes, again)

9.

Only the interviewee has been a resource officially assigned to the project, some others only at the end. There should have been someone assigned to the project on 25% at each site. However, the contact with people at these sites has been good, and they have given feedback on a need basis (and reported their worked hours as "process improvement" or something else fairly general. But your documents are not reviewed properly if the reviewer is not assigned officially and can report time on the project. The project has by that been slower, and consumed more man-hours than what would otherwise have been necessary. However, the project has reached the same result as otherwise, being delivered on time after 6 months of work (within two weeks at the time of the interview). Some reflections on the process are found in D_{F3a} [25], chapter 5.

10.

It is about to be shipped. Training with users has started, and there are two general attitudes, illustrated by the two questions "how much new do I have to learn?" and "what new support will I get?" Mostly this is a matter of personality – the negative thing is that there is a change. No one has asked for a feature that is not there, which must be seen as a success.

11.

Do not rush out and buy a system. (Superman-cape again.) First analyze needs and studies of how the existing systems are used, and then decide about the new system. Now we have a good end result, because we focused on the process and the needs beforehand: problems, needs, processes, and then we looked at systems and solutions. First one need to understand the current state – an important part of the project was to model the processes of the existing systems and put them on paper. This was the first time they were properly documented in this respect (in order to be able to discontinue them). Also, there must be an openness for other solutions than "our own current way of doing things" – still, the major reluctance to change is that it is a change, not what it results in.

12.

Not applicable?