Software Engineering Projects May Fail Before They Are Started: 
Post-Mortem Analysis of Five Cancelled Projects

Jarmo J. Ahonen\textsuperscript{a,}\textsuperscript{*}, Paula Savolainen\textsuperscript{b,}\textsuperscript{a}

\textsuperscript{a}School of Computing, University of Eastern Finland, P.O.Box 1627, 70211 Kuopio, Finland
\textsuperscript{b}Lero — The Irish Software Engineering Research Centre, University of Limerick, Ireland

Abstract

\textbf{Context}: Software project cancellations are often caused by mistakes made during the project, and such cancellations make a strong economic impact. We analyzed five cancelled software engineering projects. One case was an internal product development project of a company that sells products to its customers. The other four cases were different software engineering projects, and outcomes of these projects were planned to be delivered to external customers.

\textbf{Objective}: This study reports a post-mortem analysis of five software engineering projects with the aim of providing more knowledge about the reasons for cancellation decisions and the causes behind those reasons.

\textbf{Method}: The research method is case study. A method for a document-based post-mortem analysis was developed and post-mortem analysis was performed. All project documentation was available for analysis.

\textbf{Results}: The reasons for the cancellation decisions were well-known ones. In four cases of five, the outcome of the project was to be delivered to an external customer, but in these cases the causes of the cancellation reasons were not found from the normal project documentation. In these cases the cause of the cancellation originated in a phase before the start of the project and therefore the project was doomed before it was started.

\textbf{Conclusion}: It is reasonable to suggest that a remarkable portion of project cancellations are due to mistakes made before the project is started in the case of contract-based software engineering projects.

\textbf{Key words}: Software engineering, Project cancellation, Project failure, Post-mortem analysis, Customer, Supplier

1. Introduction

A cancelled software project is usually an unwanted situation which means loss of economic resources, despair, and embarrassment. A large cancelled software project may ruin careers and even exterminate companies. Software development history has numerous examples of project cancellations as well as consequences of software project cancellations. Unfortunately, it is likely that there are no easy means to avoid or reduce software project cancellations.

The economic impact of project cancellations is difficult to measure in any meaningful way, and even the percentage of cancelled projects is not clear (Glass, 2005). A project cancellation, sometimes called an abandonment, is a situation in which practically nothing, or even nothing at all, is salvaged from the project. A project cancellation is something that nobody wants to flaunt, and therefore getting an even reasonably accurate estimate of how many projects are cancelled is next to impossible. Some estimates have, however, been presented. For example, Charette (2005) estimated that 5-15\% of all large-scale software projects are cancelled in the USA, and

\textsuperscript{*}Corresponding author.

\textit{Email addresses:} jarmo.ahonen@uef.fi (Jarmo J. Ahonen), paula.m.savolainen@uef.fi (Paula Savolainen)
that the total yearly cost of cancellations may be as much as US$75 billion.

Assuming that Charette’s estimates are correct, the number of cancellations is daunting and their economic impact significant. The motivation of this paper is to investigate ways to reduce the number of cancellations. The basic approach to achieve that goal is obvious: if we do not conduct post-mortem reviews, we are unlikely to understand why our projects fail (Cerpa and Verner, 2009). Analysis of cancelled projects enables us to modify and improve the software development process (Reel, 1999) and to identify critical decision points before and during the project execution.

The way to avoid past mistakes is by understanding what went wrong and how it could have been avoided. Good answers to these questions regarding cancelled software projects are not, however, generally available. This makes general advancement of our software engineering project knowledge much more difficult. There are at least two reasons for the unavailability: the small percentage of projects that go through a post-mortem analysis (Glass, 2002), and the general unavailability of knowledge of cancelled software projects.

The first problem, the small percentage of projects analyzed, is not restricted to the software engineering field — some surveys have revealed that 80% of all R&D projects are not reviewed at all after completion (von Zedtwitz, 2002). In that sense, software engineers, whether practitioners or researchers, are in the same situation as other professions. The fact that the situation is not very good in other professions does not, however, give us any excuse not to perform proper post-mortem analysis.

The small percentage of analyzed projects can be improved by analysing more projects, but the second problem, the general unavailability of knowledge of cancelled projects, is a much more difficult issue to solve. It is safe to assume that the names of cancelled projects that are repeated in many articles include those cases that have been either too massive to be hidden or that have been public in some legal sense. A good example of research that uses well-known cancelled projects, some of which have been discussed in (Glass, 1999), is the one performed by Chua (2009). Most of the new cases that appear in newspapers or in scientific journals seem to fall into the same category of massive or public cancellations. Other cancellations are concealed inside the organizations, which is very understandable because neither the organizations nor the individuals involved want the details of those projects to appear in any media. The tendency to hide cancelled projects is intensified in those cases in which the supplier and the customer are separate companies.

Although some projects may be cancelled for reasons related to changes in the business environment or some other outside reasons, many cancelled projects would have succeeded if mistakes had not been made before or during the project execution. Those projects are very interesting because understanding why the cancellation took place would help us to avoid similar mistakes in the future. That understanding is especially important in order to reduce the unnecessary waste of resources.

It should be noted, however, that we do not assume that no project should fail. Failure is an essential part of high-risk projects, especially in the case of R&D projects. Although some types of projects are much more likely to fail than other types, unnecessary failures should be avoided if possible.

In order to achieve better understanding of the mistakes that caused project cancellations, we analyzed five cancelled software projects which should have succeeded. One case was an internal product development project and in the other cases the customer and the supplier were separate companies. In those cases the supplier had made an agreement with the customer for a specific project and agreed to deliver the project outcome to the customer. The aim of the study was to find out why these five projects suffered cancellation. This knowledge will help us to understand software projects better and relieve the impact of project cancellations.

The study reported in this paper was possible because of the unusually rich sets of project data that each case provided for research purposes. Each case allowed us to cut into the body of the deceased project, the body being the paperwork that includes all types of official and unofficial documents related to the project. In the analysis we looked into what happened in the projects, and especially into the actual problems encountered during the projects. The
analyzed cases are described in Section 3 and the analysis methodology is presented in Section 4.

In Section 5 we discuss the findings of the post-mortem analysis for each case. The reasons for the cancellations did not provide any real surprises. However, it was surprising that in four cases we were not able to find the cause of the cancellation reason from the documentation which is normally regarded as project documentation. Our inability to find the causes from the project documentation itself led us to extend the analysis to all available documentation. The results of that analysis are presented in Section 6. They show that four of the projects were doomed even before they were started, and that even the most valiant efforts of all stakeholders might not have been able to salvage the project.

Section 7 presents a brief discussion of the possibility of avoiding project cancellations in the analyzed cases. The validity of results is discussed in Section 8: these threats to validity are noteworthy but not serious regarding the results of the study.

The final section, Section 9, ends the article with the conclusion that in many of the cancelled projects the actual cause of the cancellation reason may be hidden somewhere in the actions that took place before the project started.

2. Related research and post-mortem methods

This section discusses related research, both research on post-mortem analysis methods and on the use of post-mortem analysis in order to learn from previous experiences of cancelled projects. However, such learning is quite difficult because very few results of post-mortems of cancelled project have been reported.

The concept of a post-mortem analysis is not very straightforward because it seems to be a fairly versatile tool. It can be used for the analysis of the end product of the project, the program, as in the research reported by Zhang and Iyer (2007), or for helping software process simulation in order to improve software project estimation (Aguilar-Ruiz et al., 2001). In this article, post-mortem analysis means an analysis performed in order to achieve understanding of a project that has already ended.

There seems to be a general agreement about the necessity of post-mortems (Reel, 1999; Glass, 2001; Birk et al., 2002; Ewusi-Mensah, 2003; Verner and Evanco, 2005), but still they are quite seldom performed (Verner and Evanco, 2005). One of the reasons for this may be that learning from past projects is important but it is not that easy to learn the “hard”, non-intuitive lessons (Williams, 2004). Moreover, concern about frank analysis especially of failure creates a natural disincentive within the organization to conduct a post-mortem; it also creates apprehension in the individual preparing to take part in ones that are held (Collier et al., 1996). But post-mortems are especially important if we are to learn from problems encountered during a project (Williams, 2004; Verner and Evanco, 2005). If one does not take time to find out what happened during a failed project, for example, then one is doomed to repeat the same mistakes (Reel, 1999; Ewusi-Mensah, 2003; Verner and Evanco, 2005). It is, however, the case that most cancelled projects are not analyzed at all (von Zedtwitz, 2002).

In order to make people more willing to perform post-mortems, the post-mortem process should be well defined (Collier et al., 1996). Fortunately, there are some reasonably detailed descriptions of post-mortem process in e.g. (Tiedeman, 1990; Whitten, 1995; Collier et al., 1996; Collison and Parcell, 2001; Birk et al., 2002). All these processes are somewhat different but they have the same general structure, which can be simplified into four phases:

1. Data collection, of which there are two basic variations. In both, data are collected from team members, and the variation lies in the utilization of project documentation. Some processes use it, whereas others do not. The data collection can be performed by interviews and questionnaires, or a combination of the two.

2. A workshop meeting, in which at least some of the people who participated in the project are present. It can consist of different types of discussion or more formal analysis methods. During the workshop, various techniques such as structured discussions, causal maps, and fishbone diagrams can be used to elicit tacit knowledge from the participants.
3. The analysis of the data, which can be performed during a workshop or separately. The analysis methods may include statistical methods for the analysis of metrics or some other type of suitable data, fishbone diagrams, and causal maps.

4. The last phase is the reporting and publishing of the results.

The common phases of the processes heavily depend on the participation of the project team members and the subjective opinions expressed by them. The role of the documentation does not seem to be very important when compared with the role of the workshops and interviews.

Post-mortem methods that fully utilize project documentation are not very numerous. The most comprehensive methods are the post-mortem analysis process presented by Collier et al. (1996) and the method proposed by Ewusi-Mensah (2003). Both of those approaches have their own shortcomings, however.

The process outlined by Collier et al. (1996) consists of reasonably well defined steps and enables the organization to produce fairly controlled and structured results and to use a predefined process for project post-mortems. The disadvantage of the process is that it requires the project team to be available. That may not be possible in many cases because the developers have moved to other projects and are not likely to be available, as has been accurately noted by Glass (2001).

A three-sided post-mortem process has been proposed in (Ewusi-Mensah, 2003, p. 198). It consists principally of three steps: 1) a questionnaire, 2) structured interviews, and 3) analysis of archival data. The method seems to be the most comprehensive post-mortem analysis method available, although it does not specify how to analyze the archival data. It does have, however, the same drawback as the process proposed by Collier et al. (1996), namely the requirement of the availability of the project team for interview and questioning.

Most studies on project post-mortems have used methods that require either the active participation of the project team or a combination of the utilization of the project material and the active participation of the project team. In our case, the members of the project team were not available, but we had rich sets of project documentation, which were as complete as could be expected and included a variety of additional documents. In addition, our aim was to find out the actual cause behind the cancellation and that means causal analysis. Reported studies on causal analysis of projects have used fishbone diagrams, causal maps, and analysis of answers to questionnaires, in order to find out the causal relations. In all those methods the participation of the members of the project team is necessary.

The unavailability of the original project teams, the reasons of which are explained in Section 3 and in the Appendix, made all those post-mortem methods unsuitable for us. In order to perform the analysis we developed a post-mortem analysis method that is suitable for the causal analysis of rich documentation. The method is described in Section 4. In the following section, the data and the background of the data are described in more detail.

3. Data available for research

3.1. Data Sources

The number of software project cancellations available for scientific research is unfortunately small. That can be explained by the fact that companies are reluctant to tell the outside world about their failures, and by the fact that over 70% of organizations do not keep records of cancelled IS projects (Ewusi-Mensah and Przasnyski, 1995). It is, however, very important to analyze and understand real-world cancellations in order to be able to avoid repeating the mistakes that someone has made before. Therefore, the analysis of cancelled projects should be performed in as many cases as possible.

We had an opportunity to analyze almost complete project documentation of cancelled software engineering projects. Normally, any type of detailed data on software engineering cancellations is strictly confidential and very difficult to get access to. In these cases one of the authors had been involved in the case either as an employee of the software supplier or the customer, or as a consultant hired by the supplier or the customer. The type of involvement is described separately for each case in the Appendix.
The possible impact of the involvement on the validity of the research is discussed in Section 8. The availability of the detailed data is due to this involvement and to the benevolence of the companies involved.

The projects we analyzed are outlined in as much detail as possible, although the organizations involved do require anonymity in order to allow us to report anything at all. Therefore, it was necessary for us to omit some interesting information from the descriptions.

The data includes all existing software engineering process documentation such as technical documents, project plans, minutes of meetings, emails related to the project, and different types of memos. The available documentation includes everything that can be reasonably expected to be found after a project has been cancelled. In some cases the documentation includes additional information such as emails kindly turned over for research purposes. In addition to the normal project documentation, each case included a specific set of documents that were created to help in making the decision whether to continue the project or cancel it. The authors were involved in the cases during the evaluation of the project and as part of the team which created the report that was to be used when the customer made the cancellation decision.

An unfortunate side-effect of the type of cases is that the companies are not willing to admit officially that the projects existed in the first place. Additional interviews would have required the permission of the management of each company. Such permissions were requested but not given. Interviewing individual persons without the permission of the management would not have been ethical even in those cases where the individuals would have thought that their personal non-disclosure agreements would allow them to be interviewed. The permissions to use the data for research purposes were acquired in the beginning of the authors’ involvement. The permissions to use the data have not been revoked, but the companies require complete anonymity and impose strict limits on the details that can be reported.

The smallest set of documentation consists of over one thousand pages. In two cases, the documentation was available only in printed form, and in three cases it was in electronic form. Some of the printed documentation was converted into electronic form for research purposes.

3.2. The cases

The cases can be classified as shown in Table 1. The term ‘contractual’ is used in cases where the software development project was not done in-house. In other words, in those cases the software development project was performed under a commercial contract and the supplier company and the customer company were separate entities.

The common feature in the cases is that practically nothing of any project was salvaged. The only exception was Case D, in which the requirements documents were reused in the new project. Even in that case, the line drawn between complete cancellation and partial cancellation is not clear.

For a more detailed understanding of the cases analyzed, some of the basic numbers concerning the cases are shown in Table 2, and more extensive textual descriptions are presented in the Appendix. Some data has been left out of the descriptions and will be introduced during the analysis.

4. Research methodology

The research method used in the reported study is case study. A definition of case study research has been provided by Myers (2009, p. 76):

Case study research in business uses empirical evidence from one or more organizations where an attempt is made to study the subject matter in context. Multiple sources of evidence are used, although most of the evidence comes from interviews and documents.

The unit of analysis in this study was one cancelled software engineering project. Every project was from a different company and each company was involved in only one case, either as a supplier, a customer, or a company with an in-house development unit. Four cases out of five involved both the supplier and the customer companies, and the
Table 1: Case names and brief descriptions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type of project</th>
<th>Type of cancellation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case A</td>
<td>Contractual tailoring of an existing ERP system for the customer.</td>
<td>Complete. The supplier paid compensation to the customer.</td>
</tr>
<tr>
<td>Case B</td>
<td>Contractual redesign of the complete software and systems architecture for the customer.</td>
<td>Complete. The supplier reimbursed most of the costs to the customer.</td>
</tr>
<tr>
<td>Case C</td>
<td>Contractual creation of a new tailored system for a customer to replace an existing one.</td>
<td>Complete. The customer paid the bills but did not use the new system at all.</td>
</tr>
<tr>
<td>Case D</td>
<td>In-house product development project aiming to replace the existing family of software products.</td>
<td>Partial. The project was discontinued and a new project was started. The requirement engineering documents were reused in the new project.</td>
</tr>
<tr>
<td>Case E</td>
<td>Contractual product development commissioned by a few major customers.</td>
<td>Complete. The cancellation resulted in the bankruptcy of the supplier.</td>
</tr>
</tbody>
</table>

Table 2: Basic numbers.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Case A</th>
<th>Case B</th>
<th>Case C</th>
<th>Case D</th>
<th>Case E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned duration</td>
<td>12 months</td>
<td>8 months</td>
<td>14 months</td>
<td>18 months</td>
<td>17 months</td>
</tr>
<tr>
<td>Realized duration before cancel-</td>
<td>10 months</td>
<td>9 months</td>
<td>15 months</td>
<td>14 months</td>
<td>24 months</td>
</tr>
<tr>
<td>lation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planned effort (effective working days)</td>
<td>560</td>
<td>280</td>
<td>3570</td>
<td>7800</td>
<td>5500</td>
</tr>
<tr>
<td>Realized effort (effective working days)</td>
<td>&gt; 600</td>
<td>&gt; 320</td>
<td>&gt; 4000</td>
<td>&gt; 6100</td>
<td>&gt; 7350</td>
</tr>
<tr>
<td>Total team size (full and part time involvement)</td>
<td>14</td>
<td>11</td>
<td>22</td>
<td>42</td>
<td>23</td>
</tr>
</tbody>
</table>

story of the cases is the story of the organizations involved, which is a common situation in case studies of projects (Myers, 2009, p. 76).

The analysis of the cases was performed in two phases. The time elapsed between the phases was a few years in some cases. The delay between the first analysis phase and further analysis was required by the companies that agreed to cede the data for research purposes.

The first phase consisted of the evaluation of the project before the cancellation decision, or after the cancellation decision when the customer was considering the possibility of disputing the results of the project. One of the authors was a member of the evaluation team in each case.

The team analyzed project documentation and interviewed people involved in the project. The team produced a report that was fundamental in the customer’s decision making: it formed part of the project documentation used in the second phase of the evaluation. Although the first phase was conducted for other purposes than academic research, the overall research methodology is within the case study paradigm.

The second phase of the analysis was performed by the authors. During this phase, the whole project documentation was analyzed including reports created during the first phase. The re-analysis of documentation was performed in order to gain new insights and avoid possible interpretational or reasoning errors made during the first phase. The analysis was performed as described in this section.

The second phase was based on the available documentation. Interviews could not be conducted because the authors did not get permission to interview the people involved. However, some parts of the documentation that were difficult to understand, ambiguous or missing were clarified by contacting some
people who participated in the project from either the customer’s or the supplier’s side. Not all inquiries were answered, but the answers that were given provided clarifications for some specific details.

In our case there were none of the normal difficulties in getting access to the relevant documentation. The available documentation was rich and allowed many different types of analysis to be performed. It was authentic and credible, and there was no doubt of the representativeness of the documents. Since information gathered from analysis of project records and documents has been shown to be invaluable in getting at the “hidden” agenda that may be at the root of the some of the failed development projects in organizations (Ewusi-Mensah, 2003, p. 199) it was possible to trace the root cause of the cancellation reason.

It is not plausible to assume that the root cause could have been found by interviewing people (Glass et al., 2008). All of the cancelled projects were so sensitive that it would have been very difficult to find persons who would have been willing to answer inconvenient questions (Myers, 2009, p. 127). That reluctance is amplified by the fact that a company which might someday end up in court is wary about calling attention to the way it captures information about its failures (Collier et al., 1996).

Since we could research almost the complete documentation of the projects, it was possible to triangulate the analysis by cross-checking each interpretation from several documents. This was possible because there were many documents related to the same subject, and in many cases individual documents had been created by different individuals.

We used several approaches to analyze the documentation. The number of pages was so huge that we had to make some classifications, which are explained below. We identified critical incidents that were deemed by the researchers to be extremely important and pertinent to the study. We also identified series of project management and project execution events and organized them by chronological chain of events in order to find causal explanations of critical incidents.

We analyzed the cases using the documentation of each project in the following way. (It should be noted that the document analysis was a step-by-step process and the results were cross-checked after each step.)

1. All documentation was classified into the following categories:
   - Documents that outlined the actual aims of the project.
   - Administrative documentation of the project, e.g. minutes of meetings, memos.
   - Technical documents such as program listings.
   - Other documents, e.g. emails.
   - Documents that were directly related to the actual cancellation decision.
   - Other types of documents.

A single document might belong to several categories.

2. The documentation was classified according to the general structure of a software engineering project. The phases used for the division were:
   - Requirements
   - Design
   - Coding
   - Testing, and
   - Project management.

The classification was done in order to make the amount of documentation more manageable, not to assume that the project had proceeded using the waterfall model.

3. The documents left unclassified during step 2 were classified into two extra classes, which have been described by Haapio and Ahonen (2006) and Arto et al. (2008):
   - Pre-project
   - Post-project.

4. The project was reconstructed from the categorized documents by anchoring individual documents and their contents to the timescale of the project.

5. The incidents and issues that were considered by the authors to have a significant impact on the cancellation of the project or that were otherwise considered interesting were identified and analyzed.
6. The cancellation decision, its documented reason, and the root cause of the reason were traced from the documentation.

The completeness of the material allowed us to analyze the projects in detail to try to get a deeper understanding of what went wrong. First we identified the actual decision and the documentation associated with it. Then the reasons for that decision were sought and we followed the project backwards in time in order to search for the actual causes, the root causes, behind the reason for the cancellation decision. In some projects, extracting the cause of the reason was not as easy as expected.

5. Post-mortem analysis of five software project cancellations

5.1. Analysis of the cases

The categorized and classified documentation was analyzed according to the document classes defined in Step 2 of the methodology described in Section 4. The first class of documents is the requirements class, i.e. those classified as belonging to the requirements phase. From this documentation, all interesting incidents and issues were extracted. This was performed for every phase from Requirements to Project Management.

The incidents and issues belonging to the requirements phase are shown in Table 3. It was interesting to note that Case A, Case C, and Case E had, in principle, very clear and precise requirements available from the beginning of the project. In Case E there was, however, no documentation that would have shown that the supplier could understand the technical demands of the requirements. The general vagueness of the requirements phase of Case B was intensified by the poor project management and the selected way of working. Case D encountered the common requirements engineering problems of the unavailability of domain experts and a too strict timetable.

The incidents and issues belonging to the design phase are shown in Table 4. In Case A the architecture of the new system was based on the existing system, which was to be somewhat tailored. The design of the tailoring turned out to be very difficult because the architecture of the existing system did not enable all the functionality required from the final software. In Case B the vagueness of the requirements affected the architecture design very badly, which was worrying because the main aim of the project was architecture design. In Case C there were no architecture problems, but the strict timetable made it necessary to drop several features. Case D was a conventional technically demanding project which required creative solutions. In Case E it was clear that the supplier had not grasped the whole meaning of the requirements, presumably due to a lack of theoretical knowledge.

The incidents and issues belonging to the coding and testing phases are shown in Table 5. Case A was suffering from serious problems during the coding phase due to the extensive changes required and the technical difficulties those changes presented. No testing was ever performed. In Case B neither coding nor testing was present because those phases are...
Table 4: Incidents and issues belonging to the design phase of the project life-cycle.

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case A</td>
<td>Architecture based on the existing system.</td>
</tr>
<tr>
<td></td>
<td>- Serious architectural changes required to the system although they had not been expected.</td>
</tr>
<tr>
<td>Case B</td>
<td>Architecture design was poorly documented and poorly communicated to the customer.</td>
</tr>
<tr>
<td>Case C</td>
<td>Many features dropped due to the too tight timetable.</td>
</tr>
<tr>
<td>Case D</td>
<td>The overall complexity of the architecture was much greater than previous experience led to assume.</td>
</tr>
<tr>
<td></td>
<td>- Completely new interfaces to other systems had to be created.</td>
</tr>
<tr>
<td>Case E</td>
<td>Lacking understanding of the impact of the requirements on the architecture.</td>
</tr>
<tr>
<td></td>
<td>Lacking theoretical knowledge.</td>
</tr>
<tr>
<td></td>
<td>- Agile methodology used.</td>
</tr>
</tbody>
</table>

Table 5: Incidents and issues belonging to the coding and testing phases of the project life-cycle.

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>Case A</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Case B</td>
<td>NA — architecture project</td>
</tr>
<tr>
<td>Case C</td>
<td>Missing coding documentation.</td>
</tr>
<tr>
<td></td>
<td>Communication between the supplier and the customer’s internal project team inadequate.</td>
</tr>
<tr>
<td>Case D</td>
<td>New tools were used.</td>
</tr>
<tr>
<td>Case E</td>
<td>Agile methodology used.</td>
</tr>
</tbody>
</table>

| Testing | Case A | No real tests were ever run because the system was never complete enough. |
|         | Case B | NA — architecture project                                                   |
|         | Case C | Far too buggy according to the reports by the customer’s internal project team. |
|         | Case D | The system was tested with real data and found to be much too slow for real use. |
|         |       | The architecture was deemed to be inadequate for the task.                 |
|         | Case E | The software turned out to be too buggy.                                   |

not a part of an architecture project. Case C suffered from missing or poor quality documentation. Case D turned out to be a dead-end, and Case E never fulfilled the aims set for it.

In addition to the incidents and issues belonging to specific phases, project management-related extraction was also performed. The results of this analysis are shown in Table 6. The project management-related incidents and issues were not unexpected: most of the projects suffered from some problems associated with project management. The only exception was Case E, in which the project management had clearly been competent.

5.2. Reasons for project cancellations

The cancellation decision in Case A was made during the coding phase and before the project had exceeded its schedule. The customer started to question the possibility of the overall success of the project, and suspended it for a while in order to perform an evaluation of the situation. In that evaluation, the technological foundation (the ERP system that was used as the basis of the tailored system) was discovered to be inadequate for the task due to the architectural solutions used in the system. The customer decided that there were no options other than cancellation. After the project had been cancelled, the supplier paid compensation to the customer.

In Case A the actual cancellation decision was made after the customer had evaluated the situation. According to the evaluation, the possibility of getting a satisfactory result out of the project seemed to be unachievable in every sense regarding costs, schedule, and technological suitability. Although we went through the project documentation belonging to the phases preceding the cancellation decision many times, we were not able to identify the actual cause of the cancellation reason. The question why the system to be tailored was selected in the first place remained a mystery.

Case B was practically an open-ended venture. The customer understood the need to create a new architecture but was not able to define the target precisely. Therefore, the responsibility of the supplier was significant and the customer had to trust the supplier. The costs and the lateness of the project led
the customer to evaluate the results already achieved, and the results were found to be unsatisfactory. The customer was dissatisfied with the project and saw no reason to continue with it. The supplier paid almost complete reimbursement to the customer.

The decision to cancel the project in Case B was made during the design phase, and the reason for the cancellation was the poor quality of the results combined with the overrun schedule and costs. Although the reason for the project cancellation was clear from the material, the cause of the problems identified as the reason for the cancellation could not be identified from anything that happened during the project. The question why there were problems with quality and the schedule remained open.

The project in Case C was plagued by great haste. Although haste was present during the whole project, it was completed almost in time and almost within budget. However, the result was never used: it was not the software that the customer had wanted, due to poor quality and missing features. The customer’s management decided not to use the software. The customer did not initiate any actions against the supplier.

Since the project was not continued in order to improve the software, the project can be considered cancelled. The decision was made after the project was, in a sense, completed, and the reason for the cancellation was poor quality and missing features. Overall haste marked every aspect of this project: for example the customer did not have enough time to approve prototypes, and several features were dropped in order to keep the project on time. The question why a project with too strict a timetable was allowed to start at all could not be answered by reading the actual project documentation.

Case D was a clear example of an internal product development project. Due to inappropriate technical decisions, the architecture of the developed software was not capable of doing the intended task. When the software was tested with real data it was found to be too slow for the intended real-time use. The higher management of the company decided commission an evaluation of the situation and after that evaluation it was decided that the only way to create the product was to drop the project and start a new one. The cancellation decision was made during the combined coding/testing phase after the first prototype was tested with real data. The reason for cancelling the project is one of the technical reasons for terminating research and development projects discussed in (Kumar et al., 1996). The cause of the cancellation was the unsuitability of the selected architecture for the situation, and this was not noticed before the system was tested with real data.

The reason for cancelling the project in Case E was a combination of poor quality, exceeded budget, and exceeded schedule. Further analysis of the perceived poor quality showed that the software was not technologically up to the use for which it was intended. After several buggy versions had been produced and the schedule had been exceeded, the customers lost patience and decided to drop the project, which caused a disastrous impact on the supplier, leading to bankruptcy. The cancellation decision was made during the testing phase of an iterative development cycle.

Table 7 shows a summary of the problems encountered during the project, the cancellation decision, the reasons for the cancellation, and the possible cause of the cancellation. The table has an additional row showing whether there were features missing or quality problems in the project. It is notable that in only one case could the cause be identified as something that had taken place during the project. In other cases we were not able to identify something that had taken place during the project as being the cause of the cancellation reason.
Table 7: Problems, decisions, reasons, and causes.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Case A</th>
<th>Case B</th>
<th>Case C</th>
<th>Case D</th>
<th>Case E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>XR</td>
<td>XDR</td>
<td>X</td>
<td>XC</td>
<td>X</td>
</tr>
<tr>
<td>Coding</td>
<td>XD</td>
<td>X</td>
<td>XD'</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>X</td>
<td></td>
<td>D'R</td>
<td>XD</td>
<td></td>
</tr>
<tr>
<td>Project management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Poor quality and/or</td>
<td>R</td>
<td>D'R</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>missing features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X = problems, D = cancellation decision, R = reason given in the cancellation decision, C = cause of the reason.

1In Case C the cancellation decision was made after the project had ended.
2In Case D the decision was made during prototyping with real-world data.

The reasons for the cancellation decisions included poor quality, exceeded schedules, exceeded costs, and technologically inadequate software. These reasons do not provide any surprises, but it was surprising that the question why those failures happened was very difficult to answer in any meaningful sense when we considered only the project. Only by tracking the cause of the cancellation reason backwards in time were we able to identify that cause. However, we were able to identify the cause of the cancellation reason only for Case D, in which the selection of an unsuitable architecture made the actual cancellation reason (the slowness of the system) manifest itself later.

In other cases it was not possible to identify the cause of the cancellation reason from the normal project documentation. Therefore we had to expand our analysis to cover all project documentation. The results of that analysis are presented in the next section.

6. Doomed to fail or what happened before the project started?

6.1. Causes of the failures

In the previous section we analysed the cases and noted that the root cause of the cancellation decision was identified only in Case D: other cases did not provide relevant explanations. In this section we present the other cases and discuss the causes of the reasons for the cancellation decisions.

When trying to identify the causes of the cancellation reasons of other four cases, we summarized our observations thus:

1. the cause that made the reason for the cancellation decision valid seemed to exist from the beginning of the project;
2. we were not able to discover all the important reasons for the project cancellations by looking at the phases of the project; therefore
3. the cause had come into existence before the project started.

These observations made it clear that the cancellations of these cases cannot be sufficiently explained by something that can be clearly identified as belonging to a specific project phase. The conditions responsible for the actual cancellation reason seemed to be present in the project from its beginning, and therefore not explainable by anything that took place during the project itself. Consequently, we took a closer look at the part of the project documentation that had been classified as belonging to the pre-project phase in order to find out what happened before the project actually started. The events of each case are briefly described in the following paragraphs.

The project to tailor an existing system to fulfill the requirements of a competent and knowledgeable customer, Case A, turned out to have design problems that could not be expected to be solved in a reasonable time and with reasonable costs. Those problems were the reason for the cancellation decision, although problems with coding and project management were also present. However, the fatal problem — the unsuitability of the architecture of the existing system — lay behind the other problems. The project was doomed from the start, but why?

The customer had created a detailed description
of the target system before contacting any supplier. That description consisted of process descriptions and detailed specifications of the required functionality. After creating the documentation, the customer made a survey of the market and contacted promising-looking suppliers. The customer started negotiations with one supplier and requested a detailed tender.

After comparing the requirements provided by the customer and the issues covered in the tender, it was clear that the tender had been created without any real understanding of the material provided by the customer. Some of the critical and very complicated requirements provided by the customer were not explicitly covered in the tender, which was created in a way that assured the customer that the requirements would be satisfied with a reasonable amount of changes to be made to the existing product sold by the supplier. The agreement was based on the tender.

The fiasco in Case A was unavoidable after the agreement had been signed. The supplier had made a mistake in making an unrealistic tender, and the project could not be saved regardless of all the efforts of the supplier or the customer. The fatal mistake was to offer a solution that was not doable in any reasonable way.

It is not possible to say exactly why the supplier in Case A offered an unsuitable solution. We have to conclude that for some reason the supplier did not pay enough attention to the requirements documentation created by the customer. If the supplier had read carefully the material provided by the customer and understood the requirements, it is unlikely that the supplier would have made the tender at all.

In Case B the project was introduced to the customer by the supplier’s representatives. The customer understood the significance of creating an overall architecture and asked several suppliers to provide tenders. The supplier that had suggested the idea was selected. This tender named several very competent people as the project team. The agreement was based on that tender.

The problems in Case B can be traced to the time between the date of the tender and the start of the project. The people named in the tender were very competent and would surely have been able to handle the project without any problems. However, when the project started, these people were not available, because they had been appointed to other projects before the start of the project. Therefore, one of the tasks of project managers, namely making sure that the right people are in the right jobs (Moore, 1999), was out of the control of the project manager. The supplier had to use less experienced people for a very demanding project. Problems were inevitable, and the cause of the cancellation can be identified as the staffing decision.

In Case C the customer had made an internal decision to replace an existing system with a completely new one. The customer created a good analysis of the target system before inviting tenders from several suppliers. Only one of the tenders was interesting to the customer, who negotiated an agreement with the supplier. In the agreement, the price of the project was the same as in the original tender, but the proposed timetable was shortened fairly dramatically because the upper management of the customer wanted the project to be completed as soon as possible.

The schedule was fixed in the agreement and was deemed doable but clearly too ambitious by the experts on both the customer and the supplier side. Although the schedule was tight, the supplier acceded to the desires of the customer’s upper management, and both parties signed an agreement with that tight schedule. We assume that the reason why the supplier agreed to the schedule was the fact that the customer did not haggle over the price quoted in the tender. It could be that the profit-oriented optimization of the supplier’s project portfolio won over caution. That is fairly understandable if we consider the optimization of the sales and delivery project portfolio in the sense presented by Tikkanen et al. (2007). However, this was a mistake, made before the project started and which made a successful outcome less likely. The cancellation of the software developed in Case C was fundamentally caused by the schedule being too tight. This is consistent with the finding of Glass (1998), who noted that the schedule is often the most serious problem.

The story of Case E is actually a tragic one. The supplier had a few major customers with whom they worked in happy cooperation. They asked the sup-
plier to extend the software for a good price, and the supplier agreed. However, that was a serious mistake because the supplier did not really know how to create the requested functionality. The supplier promised to do something technically very demanding that they had never done before. Their lack of knowledge was evident from the technical documentation and plans created during the project, and that lack doomed the project from the start. Obviously, it is not wise to agree to do something that you do not know how to do.

All four projects analyzed in this section were projects in which the customer and the supplier were separate companies. The supplier had agreed to deliver the outcome of the project to the customer. However, each project ended in a cancellation due to fatal mistakes made before the project started, mistakes that made it either impossible or very difficult to salvage the project.

6.2. Summary

The real cause of the failure in these cases was a fatal mistake made before the project started, related to the tendering, to the agreement or to something else that happened before the beginning of the project.

In our analysis we encountered four of such mistakes:

- Making an unrealistic tender or agreement due to lack of understanding of the real needs of the customer (Case A);

- Staffing decision made due to the unavailability of experienced people when the project team was selected (Case B);

- Taking serious risks by agreeing to the customer’s demands of a tight schedule (Case C);

- Promising to extend the functionality of an existing product without deep understanding of the technical problem (Case E).

The pre-project phase where all these mistakes were made precedes the project and has also been discussed in (Haapio and Ahonen, 2006) and (Artto et al., 2008). The timing of the critical mistakes in each case is shown in Figure 1. To facilitate comparison, the timing of the mistake made in Case D is also shown.

![Figure 1: The timing of the mistakes.](image)

The critical mistakes in Case A, Case C and Case E were made before signing the contract with the customer, during the tendering/negotiations phase. In Case B the mistake was made during the final selection of the project team. This could not be done before the customer had made the order, but was done before the project started. This can be considered to be the project start-up phase (Fangel, 1991; Turner and Cochrane, 1993). In Case D the mistake was made during the actual project.

The types of fatal mistakes and the pre-project phase in which they were made should make one reconsider the reasons for project failures. Even the most valiant efforts of the project manager, the project team, and the management of both the supplier and the customer may not be enough to salvage a project if a serious enough mistake has been made before the project starts.

The results of our analysis provide interesting insight into the cancellations of software engineering projects.
7. Would it have been possible to avoid the disaster, and could it have been foreseen?

Although the root cause of the project cancellation was found in every case analyzed, it is necessary to consider whether it could have been possible to avoid the project failure. In this section we consider the possibilities.

In Case A the mistake was made in the tendering/negotiations phase by proposing an unsuitable solution. In this case it would have been impossible to achieve success with the product regardless of any project management efforts. Therefore, the project could not have been salvaged after the tender had been accepted by the customer. The only way for the supplier to avoid problems would have been not to offer the project at all, or at least not to have signed the agreement.

A careful analysis of the material provided by the customer would have enabled the supplier to realize that their product was unsuitable for the intended use. Hence, the upcoming project failure would have been foreseen even before the supplier made the final tender.

In Case B the mistake was made during the selection of the project team. The possibility of successfully completing the project existed in Case B even after the project started. Although the project team was not suitable for the project, it would have been possible to correct the course of the project by changing the team before the project started or when it had been going for a very short time. When the project had been going for some time, the possibility of avoiding failure diminished. However, it is not clear that the supplier could have made any changes in the team. The available evidence suggests that more suitable people were not available during the time: they were involved in other projects, and had no time for Case B. This shows how important it is for a company that sells projects to time sales and project operations in a way that makes unwanted consequences less likely. The realization that there were problems was, to some extent, delayed by the selected lightweight project management protocol.

The importance of the timing of new projects with current projects has been discussed by Cooper and Budd (2007), and the real-world consequences of unexpected delays or other disruptions can lead to a project failure as in Case B. The supplier had only undesirable alternatives available, because transferring the team promised in the tender to the project would have endangered other projects, and selecting other people made the success of the project less likely. The supplier made the decision to use a less experienced team before the project started, and the project failed.

In Case C the problem of a too tight schedule was present from the tendering/negotiations phase. The supplier’s and the customer’s experts were well aware of the dangers that the schedule presented. The supplier’s and the customer’s management decided to take the risk and signed an agreement, and the project failed. It could have been saved by rescheduling the project, but there are always organizational issues and psychological reasons why such actions are not taken. In addition, rescheduling would have required renegotiating the project agreement. In other words, the project could not have been saved without changing the agreement: no other action would have been sufficient. In that sense, it can be said the project described in the agreement could not have been saved after the agreement had been signed.

The only in-house project, Case D, was a viable project up to the architectural decision. Thereafter, the possibility of making the project a success diminished. After a while, the amount of work spent in implementing the wrong architecture made it less likely that success could be achieved without starting afresh. The unsuitability of the architecture could not have been noticed without previous experience of similar architectures, or without testing the system with real data.

In Case E the original mistake was made in the tendering/negotiations phase. However, the project could have been saved by getting the required technical know-how from outside sources. The possibility of avoiding disaster was present until the patience of the customers wore out.

In three cases the mistake was made during the tendering/negotiations phase, in one case during the start-up, and in one case during the actual project, as seen in Table 8. The timing of the mistake has a decisive impact on the possibility of avoiding disaster.
Table 8: The time of the mistake and the possibility of avoiding the disaster.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Case A</th>
<th>Case B</th>
<th>Case C</th>
<th>Case D</th>
<th>Case E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tendering/negotiations</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Agreement</td>
<td>L</td>
<td>L</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project start-up</td>
<td>—</td>
<td>M</td>
<td>—</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Actual project</td>
<td>—</td>
<td>L</td>
<td>—</td>
<td>M</td>
<td>C</td>
</tr>
</tbody>
</table>

M = Time when the mistake was made. C = Mistake was still correctable. L = It was too late to correct the mistake after this point. — = Nothing could be done any more.

8. Validity of results

8.1. Construct validity

The common threat to the construct validity of case studies is the failure to develop a sufficiently operational set of measures, and the use of subjective judgments to collect data (Yin, 2009, p. 41). Our study was based on documentation and a few elucidating answers to clarifying questions. In each case the documentation included the complete project documentation and a large number of emails and memos related to the project. The available documentation was rich, authentic, and credible, and there was no doubt of the representativeness of the documents. In each case there were many documents related to the same subject, and in many cases individual documents had been created by different individuals. Hence, there were multiple sources of evidence and we did not need to make any subjective judgments in the data collection.

Processes for developing software are well-known (see e.g. (Royce, 1998), (Pressman, 2005), and (Sommerville, 2007)). We were able to reconstruct the projects and their internal structure by using this knowledge and available material. The reconstruction was based on the methodology described in Section 4 and the general structure of a software engineering project. Our study relies on documentation around the cancellation decision, the documented reasons for the cancellation decision, and the completeness of the available documentation.

The main methods to increase construct validity are the use of multiple sources of evidence, establishing a chain of evidence, and having the draft case study report reviewed by key informants (Yin, 2009, p. 41). The chain of evidence was complete because the material was without gaps in each case. The complete material enabled us to achieve our aim of the reconstruction of the chain of events, and therefore the chain of evidence for every case. It was not possible to have the members of the original project teams review the draft reports and review our analysis, for the reason explained in Subsection 3.1. However, due to the nature of the material, the construct validity of the study is not threatened.

8.2. Internal validity

Although internal validity is a concern for causal case studies (Yin, 2009, p. 42), that threat is not a major concern in our study. The causal reasoning of our analysis was based on the possibility of following individual events and issues backwards in time by using the reconstructed project. Our analysis of the causal relations between events and issues was supported by the software engineering literature, including software development process models, which made it possible for us to check our causal interpretations in the light of previously validated knowledge of software projects.

Because the project documentation of our cases consists of many documents concerning similar subjects (e.g. emails, memos, reports, tenders, and agreements), it was possible to compare them and check our interpretations from several documents written by different individuals. The possibility of errors in the documentation itself cannot be discounted, although the completeness of the documentation provided us with ways to make the possible impact of such errors very small. The possibility of checking the results from several documents and evaluate them in the light of well-known knowledge on software development make our logic valid.
It is, however, a threat to validity that both authors were aware of the evaluation reports produced during the first phase of the analysis, i.e., the reports that documented the cancellation reason. The influence of the evaluation report was minimized by analyzing alternative causes of individual incidents and issues. Knowledge from previous research on software engineering projects and careful analysis of the material made it possible to reconstruct the justification of the cancellation decision and therefore minimize the influence of the original report.

8.3. External validity

External validity refers to the problem of knowing whether the findings are generalizable beyond the immediate case study (Yin, 2009, p. 43). The main threat to the external validity of our study arises from the possibility that the analyzed cases are unique. Although there are several unique characteristics in each case, four cases have some common features also: one is an inherent feature of the cases and the other is that the mistake was made before the project started.

The inherent feature of the four cases is the customer-supplier relationship, and the fact that the supplier had agreed to deliver the project outcome to the external customer. In Case A, the supplier had agreed to execute a project and both modify and extend an existing system for the customer; in Case B, the project was completely customer specific; in Case C, the supplier had agreed to develop a bespoke system for the customer; and in Case E, the supplier had agreed to extend its existing software product for its major customers. All these cases were different software engineering projects but in each case the outcome of the project was supposed to be delivered to the customer. The other common feature in these four cases is that the mistake was made before the project start, in the pre-project phase. These mistakes are discussed in Sections 6 and 7.

Case D was different, being an in-house product development project. In that case there was no customer-supplier relationship in the same way as in the other four cases.

Given the foregoing, we propose that the results of our study may be generalized to those cases in which there is a customer-supplier relationship and the supplier has sold a project to the customer.

8.4. Reliability

Reliability means that another investigator is able to follow the instructions and conduct the same case study all over again, and will arrive at the same findings and conclusions (Yin, 2009, p. 45). In our study, the reliability of the results has been assured by using a well-defined analysis methodology, discussed in Section 4. The step-wise analysis was possible due to the completeness of the material. In addition, each author performed his/her analysis independently.

8.5. Biased perception and interpretation

Although the authors’ involvement in the cases made the data available in the first place, it may have had a negative impact on the validity of the interpretations. The type of authors’ involvement is described in Section 3 and in the Appendix. In each case the involvement started when the project was already troubled. The role of the author involved was to participate in the evaluation team, which analyzed the project and the possibility of making the project succeed. The authors were not involved in performing the actual project, only in evaluating it. Therefore, personal involvement does not threaten the validity of the analysis of the root causes of the cancellations. It is, however, likely that the author who was familiar with the case in the first place influenced the perceptions of the other author. That influence was minimized in the same way as the influence of the previous knowledge was minimized in the case of the author who was a member of the original evaluation team. The minimization method used careful planning of the analysis and cross-checking of the results from several documents.

8.6. Summary

It can be concluded that the threats to the validity of the study are noteworthy but not serious. This is due to the nature of the analyzed material and the selected analysis methodology. Therefore, it is reasonable to propose that a considerable proportion of software project cancellations may be caused by something that happens before the project has actually started.
9. Conclusion

In this article we report an analysis of five cancelled software engineering projects as thoroughly as possible. The detailed analysis was made possible by the availability of complete documentation of the cases. Our aim was to identify the cancellation decision, the reason for that decision, and the root causes behind the reasons, with the help of in-depth analysis of the documentation. This proved to be straightforward. The reasons could be classified as quality failures or project implementation failures, which are the most common reasons for project cancellations (Pinto and Mantel Jr., 1990).

The analysis revealed that in three cases a serious mistake, which was not foreseen, was made during the tendering/negotiation phase, resulting in cancellation of the projects. In one case, the result of analysis revealed another finding: a supplier made a mistake after receiving the order from a customer but before the project started. As a result of the mistake, this project was also cancelled. Although there were several unique characteristics in each case, the common features found were the fact that in these four cases the outcome of the project was supposed to be delivered to the external customer, and that mistakes were made before the project started, either during the tendering phase or after the supplier got the order but before the project start.

Our findings lead us to conclude that in quite a few projects it is likely that serious mistakes are made before the project has started in similar customer-supplier situations. Those mistakes may have huge economic impacts on both the supplier and the customer, and may even lead to the bankruptcy of the supplier, as in Case E. Therefore, further research on the impact of the pre-project phase, i.e. the sales/negotiations phase and the project startup phase, and on the success of software projects is required in order to evade negative results and avoid unnecessary waste of resources.

Appendix:

Brief descriptions of the cancellation cases

CASE A: Tailoring an existing ERP system for the customer

The case consists of a fairly long process in which the customer decided to replace its existing ERP (enterprise resource planning) system with a new one. The customer created a complete description of the desired situation, including detailed lists of the required features of the software, descriptions of processes and information flows between the phases of the processes, and similar very high-quality documentation.

After completing the requirements document, the customer made a survey of the systems available in the market and noted the following issues: due to the specific nature of the field of the business of the customer, no ready systems were immediately available. The customer selected two systems that seemed to be fairly suitable for further analysis and tailoring, and contacted the suppliers of those systems directly. After brief evaluation, one of the systems was selected for further negotiations.

The customer had several extensive meetings with the selected supplier. The supplier delivered a written tender to the customer, indicating that the system either already covered all the requirements of the customer or could easily be tailored to cover all the missing features. The supplier said that the amount of tailoring required could be done in at most 560 effective working days.

Several months after the start of the project, the customer refused to pay further bills due to the lack of tangible results. The supplier reported the situation as well as it could, but there was a clear imbalance between the hours reported and billed, and the results. In addition, it turned out that the supplier had severe problems with its own technology. During that time the supplier had reported over 600 days of work.

One month later the customer demanded that the project manager of the supplier should be changed. After the change, the customer and the supplier performed a detailed analysis of the situation and the possibilities of completing the project in any reasonable way.
After the analysis, the supplier and the customer arranged a seminar during which the most critical features not already present in the system were evaluated. At the same time the supplier’s project manager estimated that the remaining tailoring would require several times the originally estimated amount of work. It was also realized that the system did not provide even the already promised features that were required by the customer.

One of the authors was part of a team that was commissioned by the customer to arrange the seminar and evaluate the project in order to consider the possibility of continuing it. Before and during the seminar, several people from the supplier’s and the customer’s project teams were interviewed in an unstructured way. The interviews were done to find out whether the project teams considered a successful completion of the project possible within a reasonable budget and time. In addition to the interviews, the project was evaluated through existing documentation including emails. After analysing the interviews, the seminar as a whole, and the documents, the external team prepared a report in which they estimated the alternatives. The unanimous opinion of the team was that completing the project with reasonable costs was not a viable alternative.

The customer decided that the agreement should be broken and the project should be cancelled. The supplier paid compensation to the customer.

CASE B: Redesign of the complete software and systems architecture for the customer

The aim of the project was to redesign the complete software and systems architecture of a large company. The project was a part of a large set of projects aiming at the overall replacement of the existing information technology infrastructure of the customer. The idea of the project originated from the supplier’s consultants.

The customer invited several companies to tender for the contract. The invitation was fairly general and difficult to understand. Several companies submitted tenders, one of which was chosen mainly due to the long partnership between the customer and that supplier.

The customer invited several companies to tender for the contract. The invitation was fairly general and difficult to understand. Several companies submitted tenders, one of which was chosen mainly due to the long partnership between the customer and that supplier.

Because of this partnership it was not considered necessary to sign a detailed mutually reviewed agreement. It was decided that the project would be defined and clarified during its run. In addition, both the supplier and the customer decided that “a light-weight” project management protocol would be used, and that there would be no real need for a specific steering group or a detailed project plan. One of the reasons given for the light-weight protocol was the excellence of the supplier’s project group, although several of the very competent people mentioned in the tender were not available for this project.

The project proceeded as a series of workshops, and the costs of the supplier’s work were billed four times a year. Almost a year after the project started, the customer requested some concrete results. At that time the supplier had billed for about 320 effective working days.

There were no concrete results, which was due to the lack of a project plan and to the selected way of working. Most of the results were buried inside the workshop hand-outs and memos. The customer was unable to distill the necessary results from the material, and the supplier was not able to provide the results in a single report and design. The customer made a complaint regarding the way the supplier had managed the project and the results of the project.

The customer and the supplier then decided to evaluate the project. The supplier initiated the evaluation, which was carried out by experts from both the customer and the supplier. None of the experts had been involved in the project during its execution. One of the present authors was a member of the evaluation team. The team considered the results of the project to be inadequate and the effort wasted. The evaluation team provided the same report to both sides.

The project was cancelled, and the customer and the supplier agreed on compensation.

CASE C: The creation of a new tailored system for a customer to replace an existing one

The customer decided to replace its in-house built MIS (management information system) application with a brand new one. After analysis of the requirements of the system, the customer evaluated the options: an ready application or a tailored one. After
some analysis of the existing applications it was decided that the system should be tailored.

Several suppliers were asked to provide a tender. Only one of the tenders was interesting to the customer, which negotiated an agreement with the supplier. The extent of the project was estimated to be about 3570 effective working days. The customer's upper management wanted the new system in one year, but the customer's internal experts considered that it would be impossible. The supplier's technical experts considered that the timetable was very challenging but doable. After internal discussions, the supplier agreed to the tight timetable. The project planning was started after the agreement was signed.

It was decided that the application would be developed iteratively and using prototyping. There were constant delays in getting the customer's management to evaluate and approve the prototypes. The fourth prototype was decided to be suitable for the project. At that time the project was already over two months late, but still within the budget.

The first system provided for tentative use was much too buggy to be used. In addition, the supplier confessed that they were unable to get the developed system to properly communicate with the already existing ERP system. However, the supplier did get the interface to work after extensive work.

The supplier finished the system practically in time. Because the new system was not at all better than the old one, and even lacked some of the features of the old one, the customer considered the possibility of complaining. The customer commissioned a small outside team to evaluate the possibility of disputing the results of the project. The team interviewed the supplier’s project team, the customer’s project team, and some of the representatives of the customer’s management. In addition to the interviews the evaluation team analyzed the project documentation. The results of the evaluation were provided to the customer as a written report.

This report recommended against making a complaint because of the time-table related demands of the customer. The project was accepted one month late. The bills were paid but the customer did not take the system into use because most of the managers preferred the old system.

**CASE D: Internal product development project aiming to replace an existing family of software products**

The company decided to create a new version of its application suite. One of the aims of the new version was to integrate different applications in a clear and coherent way which would make maintenance and further development easier. In addition, the new generation would be based on a different technology from that of the old suite.

The original suite had been developed bit by bit during the previous fifteen years. Some parts were older than twelve years, and the suite was not properly integrated.

A total of 7800 effective working days were to be dedicated to the development. The timetable of the project was tight because that industry traditionally shows new products at large exhibitions, and the targeted exhibition was 18 months away, which meant also fixed deadline.

At the beginning of the project there were difficulties in creating a real project plan. The main difficulties were caused by the fact that the company had not managed such large software projects before. Accurate estimation of the amount of work was very difficult because they lacked relevant experience, and they also lacked experience in the selection of new technology to be used.

The requirements engineering part of the project did not proceed as well as expected due to the unavailability of experts from the sales and customer projects. By the time that the requirements engineering phase and the architecture plan were in reasonable shape, the project was clearly late. The time available for the rest of the project was only 6 months, including the creation of a working demonstration system for the exhibition. However, the project was continued according to the updated plans.

Four months later it was realized that for performance reasons the architecture was not suitable for large installations. But this realization came after a large prototype had been set up and the prototype had been tested with real data.

The company asked several outside experts to analyze the situation and recommend courses of action. The experts analyzed the system and its intended use, and carried out brief interviews with the
project team. The outcome was clear: there was no way to change the system in a way that would allow the project to proceed to a successful completion. One of the authors was a member of the outside team.

After it was realized that the architecture was faulty, all of the work done up to that date was abandoned. A new project was started with the same amount of resources but with a new project manager and also changes of other key persons. The requirements documents were reused in the new project, but the technical solution was created from scratch.

CASE E: Product development commissioned by a few major customers

The supplier had several customers operating multi-site systems that communicated constantly with each others. These customers used very large and expensive persistent message-passing systems in order to make the inter-system communication safe and reliable. The customers proposed that the supplier should extend its existing XML-based messaging system by implementing persistence into the system. They proposed that they could act as testers and pay for the extended functionality.

The company started to extend its system by incorporating persistence into its message-passing architecture. The first problem encountered was the fact that the company’s software engineers were not familiar with the concept of persistence or with the architectural challenges caused by persistence.

The second problem manifested itself after the company decided that it will make the extension. In order to be useful, a persistent message-passing system should be very reliable, and this requires systematic and extensive testing. The company was not very familiar with such testing and did not perform any before sending copies of the software to the pilot customers.

The customers very soon realized that the software was far too buggy to be used for the intended purpose. After several new releases and corrections, the software was still lacking both technically and quality-wise for persistent message passing. Finally the patience of the customers wore out and they commissioned an evaluation of the software by a team of external experts. The team was asked to evaluate the possibility of a rapid and successful completion of the project. One of the authors was a member of this evaluation team.

The team analyzed the architecture of the system and interviewed representatives of the supplier company. During those interviews it was realized that the employees of the supplier company did not know how to solve the problems. In addition, the architecture of the system was found to be inadequate for the intended purpose. The evaluation team reported that it was impossible to successfully complete the system in a short time.

After receiving this report the customers decided to stick with more expensive systems and to drop the new system. The cancellation of the project resulted in the bankruptcy of the supplier.

Acknowledgment

This research was partly funded by the Finnish Funding Agency for Technology and Innovation (Tekes) with grant 70011/08, and was supported in part by Science Foundation Ireland grant 03/CE2/I303_1 to Lero—The Irish Software Engineering Research Centre.

References
