LEARNING OBJECTIVES

The software systems development is a complex socio-technical process containing cognitive, managerial, organizational, and technological challenges. However, researchers are versed at addressing phenomena in isolation, e.g., within a function, role, or view. Every so often new methods or models get adopted that place our reliance on new panaceas, but different organizations have different needs and make different choices. They make decisions about core issues, such as product architecture and concept, and project schedule, staffing, and strategy. After reading this chapter, the reader should be able to recognize the importance of approaching software systems development projects in their totality.

CHAPTER OUTLINE

Software systems development projects have been difficult to succeed in. The environment is dynamic, and exposed to risks that originate from many sources, including management. The project manager must assess the environment, develop a strategy and vision to succeed, communicate these to the stake-holders to buy into, and be forthcoming. An effective strategy and vision must factor in all the important variables, i.e., personnel, structure, task, and technology, and must be malleable and operationalized by various means. To investigate these, we attended two concurrent medium scale projects for the purpose of developing two nearly identical software systems for the same customer. The two...
projects performed very differently, which was not simply due to different starting positions. In this qualitative empirical study, we focused on the effects of stakeholders’ abilities, knowledge, and skills on strategy, vision, and other factors, such as the application of technology, decision making, influence, insight, leadership, learning, and project control. The managerial involvement in the projects, the problem identification and understanding, and the strategies to overcome them, were all very different, which led to a surprising outcome. The results of this research suggest that the differences in the capacity of managerial involvement and strategizing were critical for the projects’ outcome.

KEYWORDS
Strategy, software, project management, learning, knowledge, experience

... numbers do not mean anything to me... there are so many guys who should have had titles, but do not have titles... because the numbers; we are so given over to this tyranny of stats in American sports today that we fail to look at who has the greatest impact when the [NBA] championship is on the line.

Mike Wilbon, ESPN, April 22, 2012

1 Introduction
Software engineering has been a rapidly progressing discipline, but has remained human and knowledge intensive, and with many challenges, such as an inability to make estimates with certainty, as well as to learn from or recognize one’s own mistakes, or those experienced by others. More specifically, the development of software systems is a socio-technical process containing cognitive, managerial, organizational, and technological challenges. These have been recognized early on as often being difficult to anticipate, and assess for impact. For instance, discussions at a conference back in 1968 covered all aspects of software development, including the difficulties of meeting schedules and specifications on large software projects, as well as the education of software (or data systems) engineers (Naur and Randell 1969). To better understand various aspects of software engineering, there have been studies of real world phenomena, but their impact has been weaker than in other disciplines (Perry et al. 2000), and some have suggested a lack...
of relevance to practice in information systems research (e.g., Benbasat and Zmud 1999).

Empirical studies play a fundamental role in modern science, and many analyses and surveys of the software industry and its practices have been published (Sjøberg et al. 2007). Typically, one selected aspect or factor has been captured and investigated. For example, it was found that since the first wave of estimating tools, development environments have changed so little that the models and their predicted environments are still valid today (Jensen et al. 2006); that culture, personalities, or skills affect team performance (e.g., (Chan et al. 2008), (Karn and Cowling 2006), (Müller et al. 2009)); that project management skills affects a project’s outcome (e.g., Chen et al. 2009); and that systems development activities are based on alliances among the interested parties (Brooke and Maguire 1998). The problems of designing large software systems were analyzed by using a layered behavioral model, and the importance of individual talent was emphasized (Curtis et al. 1988), but without insights on how a project manager can close the gap between the capability of personnel and the technical challenge, which has been the case even in mainstream literature (e.g., Pressman 2009).

These issues motivated our research, which includes a broader interest so that we can better understand what factors affect and characterize medium scale software development projects throughout their lifecycle. Team projects are dynamic organizations characterized by four strongly interdependent variables (i.e., personnel, structure, task, and technology (Leavitt 1965)). A change of state in one variable forces other variables to adjust and dampen out the impact (Keen 1981). The variables are complex entities that comprise many factors that characterize socio-technical systems, including communication, complexity, experience, knowledge, learning, method, motivation, personal judgment, resources, risk, skills, strategy, and workplace politics. To steer projects as planned, project managers control these factors. For example, a design can be selected that will minimize the risk of specific errors; a project plan can be used to bolster communication and teamwork among engineers; and a prototype can be built when dealing with an unfamiliar domain or technology. Organizing and strategizing are constants in managerial work that utilize similar hands-on skills and techniques so as to
accomplish change or improve a respective design (Whittington et al. 2006).

We believe that these factors are best monitored \textit{in vivo} for their dynamics and effects so that one can better qualify the attributes, such as avoidable, clear, difficult, experienced, good, poor, and size, that have been used in literature (e.g., Keil et al. 2003). Herein, we present a longitudinal and qualitative empirical study that followed a software systems development program that consisted of two concurrent reengineering projects for an internal customer. Two in-house IT divisions were responsible for the projects, and other in-house divisions and teams also contributed. They all used the same managerial and quality procedures, and both projects used the same technology. From a system requirements viewpoint, the projects appeared nearly identical in complexity, scope, and size, and, consequently, the systems they were developing were nearly identical. The time frame for the program was nine months, which included the full software lifecycle and a period of two to three months for the live testing, but the initial plan for the whole program made the tasks for the two projects different. One project used its first month to recruit personnel and set up a development environment. The plan for that project was to reuse work products from the other project, which would reduce the cost of development and maintenance. The reuse plan never eventuated, but the organizational behavior and mindset due to the reuse strategy persisted after the plan changed (e.g., the supplementary knowledge transfer in which the contributing teams with dissimilar knowledge could facilitate their mutual performance (Buckley et al. 2009) remained scant and unilateral). The plan change made the projects directly comparable, and we observed major differences in the factors important for these projects. At the start of the program, the newly recruited team completely lacked application domain knowledge, but had solid solution domain competence. The other team had strong application domain knowledge and ties, but poor solution domain competence. Eventually, the former by far outperformed the latter. A project manager with both managerial and software systems background outperformed a much more experienced project manager who built his career in the environment, but lacked the software competence. In short, the outcome and the problems
experienced by the project teams suggest a somewhat different course of events and sources of problems, and their ultimate impact on work performance and quality than previously reported by researchers.

Clearly, engineers and managers alike should be ready to step in and take on tasks that require different skills at different times. They should be able to make decisions and take initiatives that may cross the boundaries of their own, strict areas of responsibility in order to help their teams out and make their projects as successful as possible. These roles do not exist in isolation but are intricately intertwined, and depend on mutual adjustments. We find that the project managers, as the leaders, played a role that ultimately affected the performance and decided the outcome of these two projects. Their different mental models, characterized by their individual belief system, observability, and predictive power towards a target system (Norman 1983), caused them to see different opportunities, strengths, threats, and weaknesses, to make a very different judgment, and to seek different resolutions for problems that occurred on their respective projects. In other words, practice is central to understanding work, and the acquisition of knowledge and social identities at work (Brown and Duguid 2001), but achieving that is circumstantial. Below, we continue with a review of the research background, followed by a description of the program environment, developmental phases, a self-assessment, and an analysis thereof.

2 RESEARCH BACKGROUND

An influential survey of industry reported that software development projects’ failure ratio stands at 68% (Standish Group 2009). The survey reflects IT executives’ judgment and perceptions regarding project measures (see Table 1). Some authors opposed the survey altogether (e.g., (Eveleens and Verhoef 2010), (Glass 2005)), whereas a recent study found that the benefits of user participation had been conditional (Batenburg and Koopman 2010). On the other hand, it has been long known that people in organizations find themselves hard pressed either to find actual instances of rational practices or to find rationalized practices whose outcomes have been as beneficent as predicted, or to feel that those rational occasions explain much of what goes on within the organization (Weick 1976). Nevertheless, it is interesting to notice
that the first three criteria in Table 1 were found outside the projects, and the next three are project management-related. The staff related criteria weighed in at only 11% (i.e., 8 + 3). A similar European study of 214 projects in 10 major sectors of economy found that only one in eight projects was successful (McManus and Wood-Harper 2008).

Table 1  The success criteria for software projects according to CHAOS

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User involvement</td>
<td>19%</td>
</tr>
<tr>
<td>2. Executive management support</td>
<td>16%</td>
</tr>
<tr>
<td>3. Clear statement of requirements</td>
<td>15%</td>
</tr>
<tr>
<td>4. Proper planning</td>
<td>11%</td>
</tr>
<tr>
<td>5. Realistic expectations</td>
<td>10%</td>
</tr>
<tr>
<td>6. Smaller project milestones</td>
<td>9%</td>
</tr>
<tr>
<td>7. Competent staff</td>
<td>8%</td>
</tr>
<tr>
<td>8. Ownership</td>
<td>6%</td>
</tr>
<tr>
<td>9. Clear objectives (goals) and vision</td>
<td>3%</td>
</tr>
<tr>
<td>10. Focused and hardworking staff</td>
<td>3%</td>
</tr>
</tbody>
</table>

Many factors, originating from different areas, can cause software development projects to fail (Charette 2005). For example, avoidable rework consumes 40–50% of developers time, business factors (e.g., competition and the need to cut costs), project size (e.g., large-scale projects fail three to five times more often than small projects because greater complexity increases the possibility of errors), sloppy development practices, and bad decisions by project managers due to tradeoffs that must be made based on fuzzy or incomplete knowledge (e.g., cost and duration estimating, and picking the wrong type of contract—all of which are as much art as science). A research goal of
this empirical study is to investigate how such factors can be anticipated, identified, and avoided or overcome in the practice of software systems development.

Deming cautions that quality is everyone’s business (Deming 2000). Organizations should institute leadership and training, and stop making too many optimistic assumptions about what personnel can do and know. The quality of collective and individual learning is a key determinant of organizational success (Hayes and Allison 1998), so that all members of an organization must be involved in continuous learning and take action to make improvements (Weldy and Gillis 2010). For example, management is not simply problem solving but related to creation (Kofman and Senge 1993), and organization design is generally an outcome, not an input (Cherns 1976). Software systems are complex and utilize various technologies and personnel with a different expertise. The need for new approaches and methods, new ways of organizing the development work, etc., has been recognized (Carstensen and Vogelsang 2001). It is possible to produce software of the highest quality and at a significantly lower cost, which is contingent, among others, on our ability to anticipate, identify, and fix problems, as well as manage change and risk (de Gyurky 2006). Capabilities and knowledge are central to strategy (Westney 1995). The incompetence of managers and other organizational behavior issues are the main reasons for excessive development cost and time, and poor quality. If the manager is the architect of both the organization and the system that is going to build the best possible product on budget and on time (de Gyurky 2006), then the management knowledge items (i.e., the estimates, quality, risk, schedules, and staffing plans (Birk et al. 1999) are too narrowly defined.

The process of software development is neither formal nor mechanical but creative and social, with learning an important part of it, which means that education, experience, and training, for example, all contribute to a person’s ability to manage risk (Hall 2003). Every human estimator draws upon his or her background of domain knowledge, education, experience, and technical knowledge in formulating an estimate (Boetticher et al. 2006), yet much IT project learning could be incorrect (Jørgensen and Sjoberg 2000). Since knowledge encompasses emergent characteristics that stem from situated and largely unplanned activities and decision-making (Suchman 1987) defining appropriate education,
experience, and knowledge is not an easy task. For example, studies of human cognition show that when people grapple with opposing insights, they understand the different aspects of an issue and come up with effective solutions (Takeuchi et al. 2008). Employees should have the ability to approach and assess objectives and problems from a different level or perspective (Applegate 1994), all of which require a creative mindset (Faltin 2007), knowledge, and skills. Knowledge determines if a design task constitutes a problem. For without knowledge there is no reuse, and knowledge is a critical resource underlying most strategies (Visser 2006). Knowledge is also acquired through experience, by working on many projects, but action and experience do not inevitably lead to learning (Jarvis 1995), since the actions and biases of individuals are selective against negative evidence (Brehmer 1980).

The most important determinant of individual differences in programming is the knowledge base possessed by a programmer, so that a programmer can be an expert in one domain and a novice in another (Curtis 1984). From the practice-based perspective, knowledge is emergent, deeply grounded in practice, and not something that can be fully captured, codified and transferred (Whyte et al. 2008). Learning in the workplace is both a cognitive and a social activity (Gherardi et al. 1998), and the key question is how new product development teams connect the knowledge flow inside and outside of the team to accomplish their innovation goal (Büchel 2007) because the division of labor/practice creates epistemic divisions (Duguid 2006). Individual developers perform many design activities, but the design of common modules, larger systems, and components is typically a highly collaborative process which involves many stakeholders who work together in teams or ad hoc groups, to discuss or find possible solutions in meetings and unplanned work-related discussions (Dekel and Herbsleb 2007). They construct a shared vision out of collected and presented data that matches their own situated experience, i.e., they increase their own and group knowledge. Human interaction is the source of knowledge emergence (Kakihara and Sørensen 2002). However, the cost in time of all these informal and planned communications can become high (e.g., Herbsleb et al. 2001), just as a lack thereof can cause possibly avoidable rework. Furthermore, when one fails to relate one’s own actions to one’s
own results, or cannot compare them to someone else’s, the personal initiative needed to make meaningful improvements cannot eventuate.

Although technologies are difficult to adopt (Lyytinen and Rose 2003), authors of previous empirical studies typically assumed that the subjects had sufficient solution domain knowledge. In our study, knowledge and learning applied equally to application and solution domains, and quality included bad design, implementation, managerial, and procedural decisions. To detect and rectify these decisions took knowledge that, on one of the two projects, either was not available, or there was no initiative, resources, time, or will to experiment and improve. For example, many people have intellectual architectures but few can translate these into actions, agreement, and creation (Grinter 1999). Learning to program is essential, but it might take up to ten years for a novice to become an expert programmer (Soloway and Spohrer 1989), and programming is so complicated that developers can work for years in a single language and still not learn all its nuances (Whittaker and Atkin 2002). If so, then one can expect that the professional who has only recently picked up a new design method or programming language cannot have sufficiently developed the knowledge and skills to perform as expected. Likewise, the project manager who manages a project that uses a new technology may not know how to adjust to the new situation. Research has shown that individual differences among project personnel (and this should be even more true in the unstudied area of programming managers) account for the largest source of variation in project performance (Curtis 1984).

3 RESEARCH APPROACH
We are familiar only with publicly available parts of the CHAOS survey (Standish Group 2009). Based on these, we conclude that there exist differences in opinions expressed by the practitioners in the survey and those by the authors cited above, which does not come as a surprise. The criteria shown in Table 1 are largely subjective, such as clear objectives and vision, proper planning, and realistic expectations. At the same time, competence, experience, and objectives do not rank as high in the survey as some researchers have suggested. The ten criteria can be interdependent, such as clear objectives and vision, focused and hardworking staff, and proper planning. Can personnel become
focused and hardworking even though their interest in the project has been partial? Our findings suggest that the answer to this question is yes, they can. Does a team that from the outside appears harmonious actually collaborate and share a mental model and vision? Our findings suggest that the answer to this question is no, not necessarily. If the computer science graduate cannot design software (Eckerdal et al. 2006), then what is the realistic performance expectation for a novice? To answer this question, the project manager can explore four scenarios: 1) a novice can only work on something simple, which is not productive, 2) a novice should be allowed more time and have a mentor, which is not cost effective, 3) a novice can become an important contributor provided a design has been selected to compensate for the lack of competence, which is preferable, and 4) a novice should be treated as any other team member, which exposes the project to risk.

The success criteria in Table 1 apply to the program, but it was not always clear whose responsibility it was and how to achieve success, because they crossed a role’s boundary. Given these, we neither wanted to start or end our study with a preconceived notion of bad and good practices, nor launch an inquiry into inferiority or superiority of a particular method or model, because every project is unique. Instead, our objective was to observe and record the ongoing flow of actions, decisions, and events, and to solicit explanations and opinions from those who were the principle actors. In particular, we investigated the importance of architecture, experience, knowledge, organization, planning, role, skills, strategy, and vision as part of the managerial influence throughout a software lifecycle. The study also sought to elaborate on the differences and similarities found in individual actors, principal organizations, and projects’ profiles, and identify the ways in which these individuals and organizations behaved, contributed, and exerted influence within the environment. Our study was conducted in a natural environment in which two very different project teams were developing two nearly identical software systems over a period of nine months. The type of occasion was highly extraordinary, and a qualitative empirical study with overt observations (see Mack et al. 2005) appeared most appropriate to provide contextually rich data and specific insights into the local perspectives of the study’s participants.
The two projects took place in two separate locations, and we could closely monitor only the project team that was located in the same city that we were. Formally, our affiliation was with that project (i.e., Project 2 or P2 for short). We learned about developments in the other project (i.e., Project 1 or P1 for short) mostly through their interdependencies. Our insight into P1 was also affected by a lack of communication and transparency between the projects. The lack of collaboration among the project teams affected our ability to collect information at its source and seek clarifications and interpretations from those directly affected or responsible. On a positive note, although Project 1 team members were not obliged to participate in our study, they were open to talk to us. In return, we limited our interviews to after hours and lunch breaks and kept them to 15–30 minutes. To compensate for this asymmetry, we asked all the actors to review our notes and summary reports. They could also remove any information they did not want to be permanently entered into our records.

The data collection was based on complete observations (Schutt 2009) and sporadic overt participant observations (Macionis and Plummer 2008) of daily routine and meetings, and analysis of secondary data, such as project documents and meeting minutes. We also used semi-structured interviews (Patton 1980), whereby individuals explained about the negatives and positives they experienced on a particular task and tried to relate them to their experience, knowledge, teaming, training, or some other relevant aspect of their project. In this sense, answers such as *I am late because he is late* were insignificant. Our visits to Project 2 were limited to once or twice a week for the first four months and coincided with only those events we wanted to attend or were invited to. In particular, we attended a team meeting once a month, two requirements elicitation sessions, nine code and design reviews, the open session of a one day interactive seminar by IONA Technologies, a meeting to negotiate the cost for an outsourced component, a demo for the customer, and a system handover session. We conducted interviews with all the individuals singled out herein, i.e., all members of the P2 team, six of the initial seven members of the P1 team, the P1 software architect, both project managers, and some personnel of the customer. After the first four months, our visits were reduced to once or twice a month.
4 EMPIRICAL STUDY ENVIRONMENT

This empirical study took place at a multinational natural resources corporation, that was ISO 9000 certified, and with strict adherence to quality procedures at all organizational levels. The workforce was stable due to the above average remuneration packages, employee stock options, and skills and technology for which there was little demand outside of the corporation, both locally and nationally. As part of their organizational learning process, employees had to prepare a detailed annual report of their work, as well as future directions and needs, for a panel of senior experts and managers to appraise and plan for information flow, staffing, training, etc. (e.g., Blackman and Lee-Kelly 2006). Seminars in core competencies were offered to permanent employees and contractors alike, and participation was encouraged.

The customer was a division that processed raw materials. The technological process had three stages, which were all similar, using chemical compounds and elements, and heat to process a batch of feedstock. Stage 1 was mandatory, and the other two were optional, depending on the characteristics of the ordered product. The stages were computerized and each was controlled by a standalone supervisory control and data acquisition (SCADA) system that was accessed via VT100 family terminals and running on a PDP family computer. Data exchange and signalization between stages was manual, i.e., via paper or phone. Operators had to type data into a computer via a keyboard, and mistakes were easy to make. Process data were stored to flat files and could be viewed in a table on a terminal or printed out for paper archiving.

The three SCADA systems had been in use for over two decades and we will refer to those as a legacy system. They were implemented in Assembler and Fortran (e.g., Metcalf et al. 2011) computer programming languages by an IT division (IT1), which was responsible for its maintenance. The five software engineers who implemented the legacy system were still with IT1. Four of them were promoted to more senior roles, and one was with P1. As the production facility and the legacy system had been upgraded over time, another IT division (IT2) and a (process) technology division (TD) were providing to the customer various IT services and control point maintenance respectively, as well as implementing additional systems, such as the computerization of the
chemical laboratory for which a standalone mainframe with a relational database (RDBMS) was installed. However, the technological process became outdated and quality of products began to lag behind market needs. Waste could not be reduced, the profit margin shrank, and the market leader position was threatened. Senior management decided on a major upgrade of the production facility, which included the replacement of the legacy system. Their goal was to build a state-of-the-art SCADA.

The new fault-tolerant and integrated SCADA systems were to run on UNIX workstations connected to a network and implemented in Object-oriented technology (e.g., Booch et al. 2007), as well as backed up by a RDBMS. Each SCADA system would have a three-tier client-server architecture, so that multiple active and passive clients could connect to it; each stage would have two UNIX workstations that would run in parallel; and each stage would have a RDBMS that could be accessed by workstations at other stages so that optimal process parameters could be selected for each batch of feedstock. Another important enhancement was the histogram for archived and live data that would replace the table and allow raw data to be displayed. New forms would be provided to input parameters so that an analysis and simulation could be performed before a technological processing started and during one.

Apart from such features enabled by the new technology, the new systems were functionally identical to the legacy system, and the current personnel at IT1, IT2, and TD had no unknowns in the application domain. In fact, their knowledge of the legacy system and technological process was intricate. In the past, as they worked on the implementation and maintenance, they regularly visited the customer for meetings and work, and got to know personally most of the personnel. The P1 team had no experience with the solution domain, because they had never before used Object-oriented technology, nor built a network enabled multitier software system with a graphical user interface. IT2 and TD were not affected by the change of technology.

The third IT division (IT3) involved in the development of the new SCADA systems was located in another city. It took about two hours to travel by car between the locations. IT3 had been working on large scale hardware and software projects for external customers, such as enterprises, and government agencies and departments. IT3 was
experiencing a prolonged period of volatility in their business, and their goal was to create new business opportunities. P2 was their first step in establishing a presence and recognition with internal clients. Since IT3 had never before worked with the customer they had no application domain knowledge, but they considered themselves experts in the solution domain. IT1 perceived IT3 as a potential rival.

The development environments for both projects were set up identically. IT1 and IT3 each purchased a UNIX workstation, and each software engineer used a Microsoft Windows personal computer to access the workstation via an interface. The customer did not participate in the purchase of neither the workstations nor the interface, and did not plan to share their new hardware with the projects. While the workstation configuration was sufficient to run a new SCADA system efficiently, the development work suffered delays and slowdowns whenever multiple engineers used the workstation at the same time to compile and execute their programs. As we report below, this became a major source of problems for the P1 team, because of the team size, and because of their inexperience with the development environment their work suffered from constant trial and error attempts.

5 TWO PROJECTS
At the time of this study, only two of the three SCADA systems were scheduled for replacement, i.e., for the first two stages of the technological process. The customer demanded that both systems should be in production within nine months or less, and the last three months should be used for the testing (Figure 1). Both projects were awarded to IT1. For the requirements elicitation, IT1 allocated two months to P1 and 1.5 months to P2. For the implementation, IT1 allocated four months to P1 and 3.5 months to P2. For the training, testing, and parallel operation with the legacy system, IT1 allocated three months, before the handover would be completed. The testing was not only concerned about the new software, but also about the new algorithms for the technological process that could not be fully tested before both systems were in production. IT1 reasoned that by reusing work products from P1 on P2, P2 would require fewer resources to implement and would keep maintenance cost down; however, the plan never eventuated. Although they were developing nearly identical systems, these two projects
represent opposite ends of the spectrum, and even this initial plan for
the program would only formally tie them together.

IT1 estimated that they lacked resources to take on both projects
and decided to work only on P1, which was for Stage 1. IT1 outsourced
P2, which was for Stage 2, to IT3, but the IT1 estimate was not disclosed.
This change of plan was of no concern to the customer, and IT1
remained formally responsible for the program. The P2 project manager
was reporting weekly progress to the P1 project manager who sat for
both projects in weekly progress meetings with the customer senior
management. The meeting minutes were never forwarded to anyone at
IT3. Instead, the P1 project manager provided feedback when the P2
project manager or IT3 senior management (i.e., the IT3 director and the
P2 project sponsor in Figure 2) asked for it, and then those were his own
comments and interpretations. We were assigned to work with the P2
team, and the P2 project manager was our main source of information
for both projects. We do not know what exactly was discussed in the
meetings with the customer. Our contacts with personnel outside the
P2 team were limited. When invited, we accompanied the P2 project
manager to open meetings and for visits. Even so, in our interactions
with personnel we tried to remain impartial and take the same approach.

Figure 1  The original schedule for the program

![Figure 1](image_url)

**Project 1: personnel and technology**

Initially, the P1 team consisted of a project manager, a software architect,
and seven software engineers (i.e., developers in Figure 2). The project
manager was an engineering technologist by education. He had 10 years
of managerial experience and 15 overall with the corporation (i.e., with
IT2). P1 was his first project at IT1 and the first project that used the
new technology for which he lacked familiarity. In our contacts with the P1 team, we learned that he was regarded as a friendly person and successful manager during his tenure at IT2, but we do not know why he decided to join IT1. He was familiar with the customer (i.e., the management and personnel), their business, and the technological process. As a manager, he was assigned an office, but he chose a cubicle instead so that he could sit together with his team. He was first to come and last to leave the office. In his own words, his job was to manage cost and scope, and his teams took care of technology and related issues.

A software architect is responsible for the organization of the overall system that is constructed from many components (Garlan and Shaw 1994). The P1 software architect had four years of experience, and used to work for IT2 before joining the project. While at IT2, he worked on a project together with the P1 project manager, who was always full of praise for his ambitious and talented young architect. Two of the seven engineers were newly recruited computer science graduates. The average work experience among the seven was seven years (i.e., 0, 0, 6, 6, 8, 9, and 22). IT1 hired consultants to support the project but still, as time would reveal, their main risk could be found in a complete lack of familiarity with the new solution domain technology. Even the graduates were simply novices who hardly did anything outside the curriculum that taught Java (e.g., Arnold et al. 2005). P1 was preceded by a five-day training session on C++ (e.g., Stroustrup 2000), a three-day session on the Object-oriented Method and UML (UML 2010), as well as a one day tutorial on the RDBMS. Later, IT1 organized a half-day seminar on CORBA (CORBA 2010).
Project 2: personnel and technology

At the time when P2 was awarded to IT3, there was no team to work on the project. P2 was estimated to be slightly smaller than P1. IT1 gave a demonstration of the legacy Stage 2 system to the P2 project sponsor. He reasoned that P2 was a small project for which recruiting two contractors (i.e., a project leader/software engineer, and a software engineer) would suffice. The P2 project leader turned manager had seven years of work experience and Corey, the software engineer turned team leader, had five. They both signed nine-month contracts. The P2 project manager had both managerial and technical experience and often played both roles on past projects. He worked in a number of business domains, such as banking, defense, healthcare, and telecommunications, but not in this one. He found management and technology inseparable, which allowed him to effectively lead by example. He was not everybody’s person but, as this project has demonstrated, he had always tried his best to deliver according to expectations and needs of his customers, and his efforts were appreciated.

Corey spent the last two years working internationally on a software project for consumer electronics with almost the same technology used on this project. Before that, he worked as software engineer for an insurance company. This was his first job to work as a contractor since returning back home. He was excited about catching up with his family and friends, which, at times, affected his work on the project.
He got carried over easily and spent a lot of time on the phone. The P2 project manager has described Corey as a solid programmer who needed reminding about his actual job. Corey was a communicative, easy going, and fun loving person who quickly got to know everyone in the office. The workspace for the two was set up as an open plan office, with six tables aligned by the window and along a corridor partition. The layout allowed them to hear, see, and speak to each other while sitting at their desks equipped with Microsoft Windows personal computers.

Two weeks into the project, the P2 project manager felt that there would be lots of liaison and managerial work for him to do, so the P2 project sponsor recruited Mary to the team. She was a computer science graduate and this was her first fulltime job. She was a bright and outgoing, albeit headstrong person with many interests. Unlike the two P1 novices, Mary appeared knowledgeable and proficient in the technology. She took on part-time work as a student to earn money for her fees. She especially liked working with computer graphics and on graphical user interfaces. During the requirements phase, when the P2 project manager was away at the customer and IT1, Corey became the team leader. Corey and Mary got along really well, but Mary’s relationship with the P2 project manager was strenuous at times. She was willing to listen to his advice and ideas, but she liked to prove that her experience was also valuable. Unfortunately, while she was an efficient programmer, her work suffered from mistakes, which concerned the P2 project manager.

**Cost and effort estimates**

These were time and materials projects, with two deliverables: a new system, and a requirements, and IT1 was also responsible for the training. Initially, the IT1 senior management made cost and resource estimates for the program according to the three phases and time frame defined by the customer. IT3 was informed about the initial P2 cost estimate when approached by IT1 with a proposal to take over the project. The P2 project sponsor did not know how the P2 estimate was made, because IT1 never disclosed a project proposal to IT3. Instead, the P1 project manager only explained basic requirements and invited the P2 project sponsor to a demonstration of the legacy system for Stage 2. After that, IT3 worked on an estimate and risks, and each IT submitted a formal proposal. Based on our insight into the two projects, we concluded that
IT1 planned to allocate each use case to an engineer for the duration of the implementation phase, whereas the P2 project sponsor guessed how much effort would be needed to reuse P1 code and how many software engineers to recruit for the project. This likely explains why IT1 believed they could not take on both projects, and why there was such a difference between the estimates.

The P1 project manager presented the P2 proposal to the customer senior management, and provided feedback that the price was too high. We do not know how the negotiations between the parties proceeded, what arguments the customer used to drive the price down, and what was the P1 project manager position. We do know that the customer had some understanding as to how much money they wanted to pay for the projects, but the source of that information was not disclosed to us, i.e., whether the customer consulted a third party or used some other source of information to figure out a ballpark. The customer did not want to pay for any risks and training due to the new technology, and argued that IT1 would be able to reuse that knowledge and skills on future projects. Then, the P1 project manager and the P2 project sponsor negotiated a new proposal. The P2 project sponsor reduced the P2 cost estimate down, but later the P1 project manager commented that IT3 did not have to comply with his requests if they had believed that the cost would be higher.

The P1 project manager estimated a higher cost for P1 because it was somewhat larger than P2, and he factored in the reuse of code by P2. The P2 project sponsor argued that with or without reuse he had to hire enough personnel because reuse was not necessarily easy. The final P1 cost estimate was about 20% greater than that of P2, but the P2 hourly rates were much higher due to the different cost structuring in outsourcing situations. Even so, with only two contractors working on the project, it looked excellent. Unfortunately, the P2 project sponsor underestimated the effort because he was not familiar with the legacy system, which appeared simple to him. His estimate was based on the brief demo and high level description of the legacy system for Stage 2, because the requirements had yet to be produced. We do not know whether this was a matter of confidence or negligence, but the IT1 and IT3 senior management did not consider the risks of P1 not being able
to deliver code as planned, and the distribution and unfamiliarity of the teams.

The objectives of the IT1 senior management were to acquire the new technology and build the system, but they underestimated the effort because they could not fully apprehend the challenge. Being the custodians of the legacy and new systems, the IT1 senior management reasoned that whatever the cost of development, they would charge for it accordingly because the technology was new to them. They believed, nevertheless, that their estimate was realistic because nothing about the new system raised concern. They would hire consultants and provide training for their personnel in new technology. There was no reason to expect the customer to ask for major or even any changes at all, because they were all familiar with the technological process and upgrades of the facility that were planned for or under way.

The IT3 senior management never expected to overrun the P2 budget. Even so, they decided to treat P2 as a fixed price project and present themselves as a competent and reliable partner. Their objectives were to build a quality system, develop new relationships, and use this project as a future reference. In hindsight, the tactic of the IT3 senior management to go about their contract as if a fixed price one did not bode well with the IT1 senior management. As problems were piling up in P1, and the P1 project manager was repeatedly asking for additional funding, it became increasingly more difficult for him to justify his requests. As one would expect, the customer senior management could not understand why P2 was performing as expected and P1 was not.

**Project 2: side project and split**

P1 and P2 were supposed to share code and design, with P1 being the supplier of common features. P1 started first and P2 was scheduled to start a month later (Figure 3). The P1 project manager estimated it would take six weeks to get their requirements ready for review and signing off by the customer, which would take an additional two weeks. At the same time, a leading technology consulting firm (TCF in Figure 2 and Figure 3) was developing a software infrastructure with common components, such as the logging subsystem and event handler, for the new SCADA. A technical leader and two software engineers worked on the project. The same consulting firm was hired to provide training
in Object-oriented technology. Twelve weeks into P1, the IT1 director terminated the TCF contract with immediate effect, and they never completed the infrastructure project that was taken over by an IT2 team. The IT1 director believed that the TCF was deliberately delaying the completion date and overcharging. Only one consultant was rehired to coach the P1 team. Later, the technical leader commented that they needed only one more day to complete the software infrastructure project, to which his audience gave a shrug of disbelief. IT3 never used their consulting services, relying instead on their own expertise.

The reuse strategy would make P2 very much driven by P1. After the P1 requirements got signed off, P1 entered the implementation phase, and both project managers started negotiating a P2 plan for the implementation phase. It soon became obvious that P1 was not progressing according to plan. Because of the difficulties, the P1 project manager asked for patience before they could sort out their problems. He neither explained what the problems were, nor presented his schedule. The P2 project manager asked his team to first look at the completed modules of the infrastructure, analyze requirements for both systems, and identify the differences and similarities in order to prepare themselves for the reuse. The P2 team also worked on a conceptual design for a new Stage 2 system because the P1 team might not be up to the task.
P1 entered the implementation phase half a month ahead of P2, but the P1 team did not have anything ready for the P2 team to look into or reuse, and they would not commit to a date or schedule. Instead, a week later, the IT1 director asked the P2 project sponsor whether IT3 could pick up a side project (SP in Figure 3) for the same customer. This was a data acquisition and display program that would run on a dedicated Microsoft Windows personal computer and was connected to a thermal probe via special hardware. This project was sponsored by TD, because they were responsible for the device (see Figure 2). IT1 had planned to work on it, but then decided that they could not do it. The P2 project manager was happy to pick up the project and keep Mary busy for two weeks. Her job was to develop a graphical user interface. She was joined by a software engineer (i.e., +1 developer in Figure 3) from another IT3 unit who worked on a software interface to the hardware. The P2 project manager (P2 PM) designed a solution, helped the developer in understanding the algorithm written in Fortran, and he managed the project. The side project also helped control the P2 cost because Mary and the P2 project manager could charge their work hours to it.

During that period, the P2 project sponsor recruited two more software engineers (i.e., 2 new developers in Figure 3) to join the P2 team because requirements revealed that the new system was much
more complex than anticipated and lots of data needed to be stored into the database. The two engineers included a database specialist and a developer with some low-level networking experience but no CORBA. The engineers joined Corey in his work on the analysis and conceptual design. The P2 project manager explained to his team his vision of the project in which members were committed to do the best they could to achieve project goals despite obstacles and time pressure. He concluded with the following remark *I see us outperforming the P1 team and crossing the finish line first!,* and the lack of progress by the P1 team supported that. The open-ended delays began to annoy the P2 project sponsor, because P2 had made no tangible progress since completing the requirements phase. While the side project eased budgetary concerns, the switch also meant that the P2 project manager and team eased their focus on the main task. The P2 project sponsor repeatedly asked the P2 project manager to accelerate work and finalize his project plan, but his answer was always *The P1 project manager wants us to wait until their issues get resolved!*

Nevertheless, the P2 project manager started preparing his team to proceed on its own. Upon finalizing the architecture for the P2 system the P2 project manager summarized the main ideas to his team and together they discussed individual work preferences and project needs. The P2 project manager cautioned his team that *efficiency was more important than effectiveness on this project.* His goals were to *build a simple, responsive, fully functional, dependable, and consistent system, within the shortest time possible, since the project was already late and more uncertainty was possible due to the time pressure and developments on P1.* To ensure his team complied, he *might challenge their detail designs and estimates.* He and Corey assisted the new members in catching up so that they could all start working together on the object design. The P2 project manager also devoted time to help an engineer with CORBA.

The P2 project manager could not further delay the project. In his opinion, the P1 team would not be able to deliver on their promises any time soon. His understanding was that the P1 plan was flat, and that all use cases would be implemented at the same time, instead of being prioritized. It was also unclear whether the prioritization of work within a use case was sufficient to enable reuse. The fact that the P1 team increased its personnel count to thirteen went towards supporting...
his opinion. Given the amount of problems that the P1 team had experienced so far, it could be expected that their code and design would have many errors, and the P2 team did not have enough personnel to deal with it. The communication and interaction patterns between the teams suggested that it would be very difficult to complete the program with any firm deadline in mind. Almost three months into P2, senior management decided that each project team should proceed on its own. The only similarity in the implementation between the two systems was the 3-tier architecture and the software infrastructure with components, because nothing else had been discussed neither before nor after the P2 implementation phase started.

Later, the P2 project manager commented that the delay was positive in the sense that the P2 team managed to prepare well for the implementation phase. The object design was detailed enough and good enough to allow a realistic and solid project plan to be constructed with very few unknowns. The daily routine was well planned for. Everyone on the P2 team was clear about the challenge. The P2 project manager added that nobody on P1 gave much thought about the object design when estimating the cost and timeframe. In his opinion, the P1 project manager was only concerned about and planned for the implementation, but he never explained what exactly the term implied. The P2 project sponsor never raised that question in negotiations either, and he never discussed with the IT1 senior management any details about the intended coordination, interaction, and sharing between the teams.

6 REQUIREMENTS ELICITATION

Due to customer’s demands, both projects were planned according to the waterfall model (e.g., Benington 1983). No project would proceed before requirements were signed off, and the system acceptance criteria was fully defined and committed to by both parties. No deployment would take place before the new system was fully implemented, because it would be neither practical nor useful for the customer to work with an increment. To install a new system, they had to shut the facility down, and that would take place during the next regular annual maintenance period during which they would be able to upgrade the facility. While there was some flexibility in the timing, because orders had suddenly picked up, it was important that project schedules fit within the original
Knowledge, Learning, and Skills in Software Systems Development

timeframe. Both projects were staffed and started as initially anticipated, but soon the management of the projects and their performance started to diverge, as the requirements phase has first revealed. Innovation is powered by a thorough understanding, through direct observation and use of targeted prototypes, of what people need and want in their lives and what they like or dislike about the way particular products are made (Brown 2008).

Project 1: requirements elicitation
The Stage 1 technological process was never standardized, and operators used various sources of data and manuals, as well as their experience and preference when processing an order. The legacy system for Stage 1 was never fully documented, and many changes to the code were made upon informal requests. All these had to be analyzed and either reconciled with the new algorithms and workflow or removed. The P1 project manager appointed Shane to lead the task and work with the customer. Shane was a graduate engineering technologist who joined IT1 eight years ago. His job had been to maintain the SCADA for Stage 1 and his good work was long recognized. Another software engineer was helping Shane. He was among the five who had developed the legacy system for Stage 1. The customer appointed their technology manager and two operators to the team. The requirements team had to produce a comprehensive requirements document and standardize the technological process. Although a demanding task, it was completed on time, but unresolved issues with the new graphical user interface for UNIX workstations remained.

Both parties were familiar with the legacy user interface and they both assumed that the contents and layouts of new forms would resemble the old forms, which served as their reference. The legacy system offered little design flexibility, whereas the new software offered seemingly endless opportunities for improvement over the old ergonomics. In their meetings, they simply used pen and paper and a whiteboard to discuss new ideas and create a hardcopy for future reference, but the process was tedious. The operators could not find a common ground when discussing details, such as the appearance, exact layout, visibility, and how to enable different functions. In fact, as we have learned firsthand during P2 requirements elicitation meetings, some operators
were not excited about the new system. Because the P1 team members had no experience with the new graphical user interface technology, they did not consider building a prototype, and the P1 project manager said that *no such provision was in the project budget and plan*. Instead, after consulting with the customer senior management, the meetings were abandoned due to *lack of progress*. It was decided that the Stage 1 operators would approve the new graphical user interface during the handover.

The P1 project manager neither engaged on the requirements nor tried to resolve the problem with the new forms. He and Shane both found that a minor outstanding issue. The P1 team believed that they thoroughly understood the mental models of the operators, and that they knew how to integrate the new system into the workflow. Furthermore, the customer told the P1 project manager that new monitors would all have the same screen resolution and size, and he developed a conviction that no monitor-specific adjustments of the graphical user interface forms would be necessary. Since the P1 project manager was not a software person, neither he nor his team considered or discussed building a version that could dynamically scale to a monitor. Instead, he relied on common sense and his perception of simplicity that would guarantee him a timely project completion, which assumed *no scalability of forms*. Generally, the P1 project manager adopted the strategy of first refusing to consider any changes to the requirements after the signoff. But when he decided to discuss a change request, he asked for additional funding by portraying it as a major change, and he instructed his team to do the same. However, as time would tell, the operators flatly rejected the new Stage 1 system because of its graphical user interface and overall poor quality.

**Project 2: requirements elicitation**

The P2 requirements phase was much more difficult. They had only five weeks allocated, but no application domain knowledge. The P2 project manager picked up the task, but he neither knew how to locate the customer nor who to talk to. He asked the P1 project manager for help and an IT1 engineer who was about to leave IT1 was assigned to the task, but he was not keen to work. After two days with lots of talk and little work done, the P2 project manager complained about the situation to
the P1 project manager who then appointed Robert to work fulltime on the task. Robert was a software engineer with nine years of experience, with seven of those at IT1. His main job was the maintenance of the legacy system for Stage 2. Robert and Shane were good colleagues, and they were both members of the P1 team. Robert was aware of Shane’s work on requirements and the difficulties that he had experienced. Like Shane, Robert was proud of his knowledge. Robert was also a good listener, open to ideas of others, and was a patient teacher to the P2 project manager regarding the application domain. He wanted to assist in building the best possible new Stage 2 system and was excited about the opportunity, as was the P2 project manager who found the whole situation challenging and new. The customer appointed the technology manager and a junior Stage 2 operator to the requirements team.

The first two meetings with the customer were difficult for the P2 project manager because he did not understand the terminology, and thus he could not engage in the discussion. Fortunately, Robert immediately took over by preparing action points for each meeting, and discussing issues. After each meeting, the P2 project manager spent time with Robert analyzing the minutes and asking about unfamiliar terms and words, and other specifics. Nevertheless, the P2 project manager realized that the main source of difficulty was the graphical user interface. No one could explain how many components could be fit into each form, what components should be used, and how to improve the current layouts, or perhaps completely redesign some of the forms. Since the new system would also serve as a knowledge management system, it was not clear how to automate the workflow and present information, and the P2 project manager certainly did not understand what the operators needed the knowledge for and how it had been archived, and used in the technological process.

The process of designing the new graphical user interface was tedious, since during each meeting there were different operators sitting in along with the division director and the technology manager. The production was organized in three shifts, and senior management wanted all the operators to have their say in order to avoid dissatisfaction with the new system and possible workers union action. It comes as no surprise that at each meeting a new set of drawings was produced on the whiteboard that had little in common with those before, and that no progress was made.
To further complicate the matter, there was a sharp division among the operators, with management unwilling to take sides or mitigate the conflict. The older operators resented the idea of having a new system. They claimed that the legacy system could be improved instead, as per their proposal. They disliked the histogram and wanted to keep the table instead. In contrast, the younger operators were enthusiastic about the prospect and wanted to use all the different components they grew accustomed to when using applications for Microsoft Windows. The constant stream of diverse ideas and the dichotomy of positions created a fragmented environment for the P2 requirements team, and the P2 project manager concluded that without a prototype of the new graphical user interface displayed in a monitor, it would be impossible to complete this important task with certainty and confidence.

The P2 project manager decided to build the prototype in Visual Basic (e.g., Lim 2012) and install it on a personal computer so that the operators could obtain a better idea about the new graphical user interface while playing with it. The prototype could only switch from one form to another, pop up dialog boxes and menus, or provide some visual feedback, such as the changing of colors, and the displaying of text messages, so as to acknowledge an operator's action. It took Mary and the P2 project manager a couple of hours to build it according to the old forms layout and whiteboard drawings. The P2 project manager never charged the customer for it because Mary was sitting idle anyway. While this was not exactly the same look and feel of the things to come, it proved close enough, and the prototype attracted much attention. Now, everyone interested could develop a realistic idea of and gain some experience with the things to come, and provide much more solid feedback on the color schema, components, font size, and layout. The older operators stopped perceiving the new system as a threat, and the young operators could see how their Microsoft Windows experience would be recreated in the new system. The two subsequent meetings became very productive. The prototype was redesigned to account for the new ideas, and the new user interface was designed, ostensibly to everyone's satisfaction.

The completed P2 requirements contained 78 pages. Six use cases were inherited from the legacy system, and updated, and two new use cases were added. The P1 requirements were reused for consistency, and...
to speed up the writing. P1 had one more use case defined and 17 more pages. However, according to Robert, the P2 requirements needed more work because all external control points and devices had to be accounted for and (re)programmed to comply with the new hardware. The task took two weeks to complete, during which a cross-functional team of 18 was assembled and worked diligently to complete the work as soon as possible. The delay did not affect the requirements review and signoff process, which took one week. Overall, all parties were satisfied with the performance during the requirements phase and the achieved outcome. As the record would show, the requirements remained stable, and the P2 team needed only additional explanations on the terminology during subsequent phases because they lacked the necessary application domain knowledge. By contrast, the P1 project manager never considered taking the same approach to finalize the P1 graphical user interface. Instead, he said that his team was not just assembled two days ago but has had years of experience working for the customer.

7 SYSTEMS IMPLEMENTATION

Both project teams in this empirical study had, arguably, mostly experienced members. The requirements for both systems were nearly identical, and realized within the same three-tiered, client-server architecture, but this is where the similarity between the two projects and systems ended. Both project teams had strengths and weaknesses, but their strengths did not necessarily facilitate their performance, and their weaknesses did not necessarily threaten their project. The behavior of a system does not depend on what each part is doing, but on how each part is interacting with the rest (Senge 1990). This applies equally to artificial and natural systems. The P2 project manager commented: The process of developing a system begins with a notion of what that system is, i. e., its main components and how to best realize and utilize them. It can be driven by many factors and require making tradeoffs. The [lifecycle] model doesn’t matter, but what one makes of it. The engineer, and more generally the designer, is concerned with how things ought to be in order to attain goals and to function (Simon 1996), and [software project] managers should be decision makers and designers (e. g., Martin 2009), and even possess the competency of an engineer.
Project 1: implementation phase

The P1 project manager engaged in cost and scope management and relied on the advice and opinion of his software architect and team in technology-related issues. They all grew accustomed to taking a functional viewpoint, and only used the requirements to create a project plan. They viewed a use case as one part of the system that was developed end-to-end by an engineer who decided on the tasks, sequence, and made estimates. This strategy worked well on past projects managed by the P1 project manager when systems were closed and integral, and required only basic programming skills. This project was different, but the P1 team was not overly concerned about the new architecture and technology and how could these affect their work. The P1 project manager repeatedly stated that IT1 prepared well for the project by hiring consultants and providing ample training to the personnel. Also, five of seven team members had previously worked together and were all good colleagues among themselves.

Initially, each engineer was assigned one use case to implement. The software architect picked up one outstanding use case, and the other one was delayed until a resource could become available or recruited. The plan for reuse by P2 prompted the P1 team to first work on the server, followed by the graphical user interfaces (i.e., the client), and the RDBMS part. The technology was new, and no one could make estimates with confidence. For example, three to four weeks were allocated to implement a (graphical user interface) form. When these ideas were presented to the P2 project manager, he commented that his team was much smaller and would not be able to complete the project on time or with any certainty. His team would be underutilized and without control...[and] would have to understand the work products, map them to Stage 2 requirements, and then modify or reuse them...[but] they could not reuse code fragments. How would teams communicate? He asked for shorter time intervals in which work products would become available in coherent and small batches, but the talks ceased because P1 was experiencing difficulties.

Those work assignments put each P1 engineer in a position to individually face constant and new challenges, which led to internal fragmentation and a loosely coupled team. Each engineer had to learn how to create and use CORBA interfaces, interact with the relational
database and map their classes to the relational model, program graphical user interfaces, and use the infrastructure. In the process, they all had to become solid object-oriented designers and C++ programmers to implement complex algorithms. The software architect was also new to the technology; but, in order to alleviate the learning needs and help the team, he made himself available to his colleagues and spent time contacting customer support and the software technology experts (see Figure 2) on their behalf. The engineer with six years of experience who had been assisting him in system related issues became the principal for the graphical user interface design, and he tried to help make forms look consistent.

The lack of consistency had two root causes (i.e., the lack of systems thinking, and the work assignments) facilitated by the isomorphic team structure. The team failed to develop a shared conceptual model of the system and did not have the will to enforce consistency due to the patchy and slow progress. The lack of knowledge and skills prevented the team from considering how to use technology to their advantage, instead of settling for the most obvious designs. They explained that "the pressure to produce made them think only about the schedule." They neither had a common approach to testing nor a consistent quality procedure. When a problem surfaced, they made ad hoc decisions, and rework was not an option. For example, some algorithms were implemented in the server, some in the client, and some were split between the two. As a result, the interface between the client and the server depended on where the use case was implemented. They never gave a demo to the customer or solicited comments, because the P1 system emerged only after all use cases were implemented.

The work assignment strategy created additional efficiency and quality issues, and multiplied risks because the lack of the shared mental model and the loose work couplings prevented the team from exploring opportunities towards efficiency, and it impaired effectiveness and teamwork. For example, each engineer took three mandatory tasks to implement the forms of a use case, which were: 1) implement forms, 2) implement a CORBA interface for the client and the server, and 3) integrate the forms and the interface. The testing either added one more task to write test code that was later discarded, or had to wait for server code and database integration. The need to acquire knowledge and
develop skills for the tasks led to poor quality and unreliable estimates because they simply did not know enough. Even more time was wasted because the tasks were exploratory instead of becoming repetitive. For example, most of them spent time on a workaround for the same problem in the infrastructure.

However, the P1 project manager reasoned that more software engineers would be needed and the team size increased to 13. The outstanding use case was picked up by a new engineer, and on some of the other use cases new two engineers worked. An engineer was allocated to work on the issue that prevented a table from receiving keyboard events. The new database analyst created queries and tables for the team, but his main job was to work on lots of process data that had to be stored, and although he was not new to the RDBMS, his training session lasted a week. The P1 software architecture emerged only after the system was integrated (i.e., before the handover). As a result, its technical value for the project immediately diminished, and its managerial value never eventuated. The same is true for the P1 client that had to be reimplemented because the forms were not scalable (see Section 9/Project 1). Unfortunately, after the projects went separate ways, our direct contacts with the P1 team became rare and the P2 project manager could only provide scant information.

**Project 2: implementation phase**

Three months into P2, senior management decided that P2 should not wait any longer for P1 to make progress, and no reuse of code and design took place except for one algorithm that was completely revised after requirements got signed off. The P2 project manager also served as the P2 software architect. P2 was also a waterfall project, but for this phase he opted for an architecture first and incremental strategy so that problems and tasks would become repetitive instead of remaining exploratory, and project complexity and risk would diminish over time by reusing components and designs. The increments took one week to complete, which was enough to implement a use case, or a testable part of it. The small milestones were desirable because the implementation phase was short and one third of it was already behind them. The P2 project manager assigned tasks as per expressed interests and project needs. Mary was responsible for the client (i.e., the graphical user
interface) and Corey for the object design and server. The other two engineers worked on outstanding server tasks, and either on CORBA or RDBMS.

The task assignments followed specializations to the extent possible, which facilitated efficiency and quality, but Mary was underutilized. It took Mary a day or two to implement the forms of a use case, while the server code took three or more days. The team reasoned that it would be hard for Mary to work on small pieces of code and design just to improve her timesheet, and they would have to brief and help her. Mary started an increment, while the server engineers were finalizing detailed designs or finishing outstanding work. Mary made many small changes while visually optimizing a form and frequently recompiled code, which ran fast because she was the only user of the UNIX workstation. When she finished her task, she worked on code and design reviews for compliance with coding standards and requirements, documentation, and on testing. As a rule, the engineer with free time ran tests and worked on issues. The mutual involvement in these activities facilitated the creation of a shared mental model and familiarity with the system, but it required a coherent, flexible, and simple design, which characterized the software architecture. In turn, the architecture facilitated the incremental strategy and sped up the work.

The P2 project manager decided to first build a communication interface in CORBA and a test-bed. This way, the client, the server, and the shadow process were integrated into a system before the implementation of use cases started, and the test-bed was yet another subsystem that reused the CORBA interface. The interface was flexible because multiple forms and subsystems could be concurrently served. The design remained coherent since no use case required its revision, and it was simple because the P2 system had only one CORBA interface with a receive and a send method, a message dispatcher based on numeric IDs, and the protocol used ASCII messages (i.e., text). For example, the server either sent a message to change the state of a graphical user interface component, or received a message to perform an action. The test-bed could display received messages for validation and send messages for execution. Multiple messages could be concatenated and sent as one without additional CORBA programming. By contrast, the P1 system had a different CORBA interface for each form, and each
method in an interface served only one graphical component in a form. One can argue that the P1 design was more by the book, but it was also inconsistently applied and inefficient both for the system and the team.

From a technical viewpoint, the P2 design improved the system performance (i.e., it was fast and responsive). For example, a complete form could be updated by a concatenated message. Messages were easy to parse because of the 3 digit ID + value syntax. The processing of a message was fast because its ID was the index into an array of registered components to find the receiver of the message. The data conversion for calculations and the data formatting for displaying were done at one time and in one place (i.e., in the server where the algorithm was implemented), whereas the P1 interface always sent raw data across. These decisions also simplified the P2 object design and further decoupled the client. From a managerial viewpoint, the P2 architecture also eliminated the risk due to the team's inexperience with CORBA, and it reduced the risk of rework by simplifying the design. The weak subsystems coupling in P2 led to decoupled tasks, which facilitated scheduling. They could overlap coding, design, reviews, and testing, and reduce the time for mutual adjustments and waiting.

The P2 strategy facilitated efficiency and quality because it reduced the number of tasks from an increment's start to a testable (sub)system. For the P2 client, it took one pooled task per increment to implement the forms of a use case. The task took one or two days to complete. Newly implemented forms were immediately tested by the test-bed, which did not require coding. Forms were dynamically scalable and they could be displayed on any monitor, so they built a client for Microsoft Windows and ran it on a personal computer, which eased the load on the UNIX workstation. Server tasks were scheduled such that it could be tested end-to-end. When implementing a use case in the server, the P2 team started with a component next to a boundary (i.e., a communication interface), and sequentially worked towards another boundary. This scheduling strategy minimized the occurrences of cases in which a task had to send feedback or request for change to a (distant) earlier task. This testing strategy could quickly detect new inherent and plausible errors and reduce the testing complexity and time.

There was always a version of the P2 system that could be immediately tested and fixed as new components were added to the
system. The P2 project manager implemented a test-bed to which Mary added a graphical user interface with various components for enabling automatic and manual testing, and for data input and output. The test-bed could also emulate the external control points and devices at Stage 2 for the purpose of testing. The test-bed was an integral part of the project because it formalized and standardized the process, and also relieved the P2 team from writing test code and the UNIX workstation from recompile cycles. It was used for black-box testing of a working version of the system and subsystems, and for checking the project status. Other decisions toward simplifying the project control, scheduling, and testing were: to implement one use case at a time and minimize work-in-process; to use small increments to create and maintain a sense of steady progress and urgency within the team that operated on a tight schedule; and to use tasks short in duration (most often only a day or two) and just enough to implement a class or a complex method or review past work. Requests for change were implemented and tested without a delay. All these created a spirit of support and a sense of mission within the team.

The flexibility and simplicity of the P2 software architecture also compensated for another weakness that was noticed on the side project (SP in Figure 3). Mary was an efficient programmer, but her code suffered from many errors, including memory leaks and performance issues. The P2 project manager used this as a motivation for the implementing the graphical user interface as a thin client. Each component and form had an ID and registered itself with a message dispatcher. Each component and form had an interface that the dispatcher used to deliver messages as events. Because the server only sent formatted data, the client could directly display them. All these made Mary’s programming work almost mechanical. Another subtle but equally compelling reason for Mary’s task assignments was that the P2 project manager wanted to teach her to appreciate discipline over impulse in her work.

Four weeks after the two projects split up, P2 had a system with two implemented use cases. The P2 project manager invited customer and IT1 management and Robert to attend a demonstration at IT3. He also wanted to demonstrate that the side project was completed. The customer did not expect this invitation, but they appreciated the opportunity to finally see concrete evidence that either project made progress. Overall, they provided positive feedback to the P2 team, which
made the IT3 senior management more confident that their plan to venture into in-house projects was finally on the right track. This was the first opportunity for the customer to see the real graphical user interface. They asked for a more vivid color scheme because their control rooms were dim and dusty. They asked for larger fonts and flash lights so that the operators could read and see them when away from a monitor. They liked the dynamically scalable forms, but they could not see its practical value because their monitors will all have the same resolution and size. The P2 project manager explained that that was how good graphical user interfaces should be built.

8 PERSONNEL AND STRUCTURE
Both projects investigated in this empirical study were undertaken for the same in-house customer, but they stand at the opposite ends of the spectrum. The P2 project manager was reporting to the P1 project manager and the P2 project sponsor on a weekly basis. The P1 project manager was reporting progress for both projects to the customer, and P2 had no formal links with the customer. The P1 personnel were familiar before the projects started, and IT1 would become the custodian of the new systems. IT1 invested in the training, and brought in consultants to work on the underlying infrastructure, to set up a development environment for the P1 team and facilitate the overall development effort. P1 can be described as a project in which both the customer and the development team had almost equal application domain knowledge, as well as complete solution domain knowledge as far as the legacy system was concerned in their respective areas of competency and responsibility.

IT3 perceived P2 as a filler project, albeit with a future potential. IT3 was located in another city and had no prior affiliation with the customer. The P2 team was formed only for the project. All the personnel were recruited by only addressing the solution domain knowledge and skills, and ignoring the application domain because that expertise was unavailable among their personnel and in the job market. IT3 had neither contacts nor familiarity with the P1 team before the project started. The P2 team members disliked the idea of visiting the customer, because the site had a dusty, hazardous, and hot environment. They complained that even the water in the water fountain was brown and undrinkable. They
saw no career value in acquiring the application domain knowledge, and they gave the P2 project manager no alternative but to become the liaison and provide the answers through his contacts.

The P2 team believed that their solution domain knowledge was by far superior, and they never felt a need to communicate with the P1 team. Their attitude was best portrayed by Mary asking *What is there to discuss with them?*—to which everyone burst into laughter. The P2 project manager tried to overcome the communication problem by introducing his team to the P1 team. He invited his team members a couple of times to join him on his regular weekly trips to the customer and IT1 during the requirements phase. He repeatedly encouraged his team to talk to their peers at IT1 and learn more about their work and the P1 system because they would have to reuse their code and design. He asked them to offer help to the P1 team so that contacts would continue. Unfortunately, these attempts did not produce a lasting result. Only Corey completed his one assignment that was essential for the P2 team. Corey set up user accounts for the P2 team at IT1 so that they could access their local information and knowledge sharing system.

Because of this negative attitude towards the application domain and the P1 team, the P2 project manager became concerned. To further complicate matters, during the requirements phase, the P2 project manager was working at IT1 and visiting the customer, and the P2 project sponsor took care of the P2 team. This gave the P2 project manager time to catch up with P1, but the P2 project sponsor negotiated cost with the IT1 senior management and recruited personnel. The P2 project manager regularly informed the P2 project sponsor about his work, but the P2 project sponsor did not reciprocate. For example, when Mary joined the team, the P2 project sponsor made Corey the team leader, and enrolled him in a two day seminar in project management that was offered by the corporation as part of their organizational learning program. However, the P2 project sponsor neither consulted nor informed the P2 project manager about the change. All these sent the wrong message to Corey who now saw himself in charge of the project. When the P2 project manager asked Corey to work on the analysis of P1 requirements, he immediately passed it on to Mary who quickly decided that *she did not understand any of the requirements.*
This situation further escalated when Mary asked the P2 project sponsor for a three-day leave. Mary was a gifted violinist and she wanted to join an orchestra for a performance. The P2 project sponsor approved her request, but he did not inform the P2 project manager about his decision. At the same time, the P2 project manager decided to build a prototype of the graphical user interface. He informed his customer about it and asked them to schedule a meeting in two days. When he returned back to IT3, he was surprised to learn that Mary had left a day earlier, and that his initiative would have to wait until the next week. That was not the kind of message he wanted to send to the customer. The P2 project manager got very upset and demanded that the team could not operate in this manner. However, Corey was not prepared to give up the power he had assumed to have. To calm the situation down, Corey remained the team leader, but was not allowed to change the project plan or task assignments without approval by the P2 project manager. To the best of our knowledge, the IT1 senior management and P1 team never experienced such problems. The IT1 director and the P1 project manager worked harmoniously together, and so did the P1 team.

A visit to IT1 took the P2 project manager a whole day, requiring a very early start and late finish to the day, but IT3 would not pay for overtime work. Instead, the P2 project manager was offered a car to make the commuting more convenient. The one way trip between the sites took about two hours by car, or three hours by intercity train and local bus. However, the P2 project manager travelled by train because he did not have a driver’s license. The P2 manager preferred to arrive 30 minutes early to prepare himself for the meeting, most often by talking to Robert. If a member of his team accompanied him, he would use that time to introduce the companion to the P1 team. The P2 project manager used the car only when a member of his team or the P2 project sponsor accompanied him, or when he invited us, but he took a nightly train back and let the driver companion return early by car. He made his visits to the customer accompanied by Robert, because Robert had a special permit to enter the site.

The P2 project manager took on the requirements elicitation so that he could get to know all the personnel involved in the project. The lack of interest by his team for interaction across a team boundary forced the P2 project manager to engage in all project issues. On the other hand, the
cost in money and time was too high to have P2 team members regularly attend meetings at IT1. On the positive side, all these factors contributed to the P2 project manager having complete control over his project and, in return, made his team happy to follow his vision. In contrast to these, the P1 project manager focused only on managerial issues, giving his team complete autonomy in technical issues. He was an engineering technologist by education, but his engineering days were long behind him, i.e., ever since he became project manager. He had no experience with or knowledge on the new technology used on P1. Neither could he engage with his team in resolving technical issues, nor understand the impact. The P1 project manager first asked his architect for his advice and opinion, and then the team discussed the issues during weekly team meetings. He only discussed issues in person with an engineer when suspecting that he was not putting in his best effort. Unfortunately, we were never invited to attend a P1 team meeting or offered access to the minutes.

To IT1, P1 was not simply a project, as P2 was to IT3. IT1 formed or utilized other teams to assist the P1 team with the technology adoption and other work (see Figure 2). Some of this work was paid for by P1 and P2, and some by IT1. The team of two software technology experts was responsible for high level issues that concerned the three new systems that would replace the legacy system, and they worked with the TCF team. Both teams were funded by IT1 and they never engaged with the P2 team. Only the experts gave a presentation to the P2 team and informed them of their role and work. The technological process team analyzed data and worked with the customer on new algorithms and methods for the processing of raw materials. That team was funded by the customer and IT1, but both P1 and P2 had to pay for their requirements review during the signoff. There were two more teams, one with IT1 (i.e., Team A) and one with IT2 (i.e., Team B) that were responsible for the implementation of the outsourced components for both P1 and P2. The P2 team was never involved in the decision-making process. Instead, when Robert learned about a decision, he informed the P2 project manager about it, because he was the liaison.

P2 used only two staff for all the interaction, i.e., the P2 project manager and Robert, whereas the P1 project manager appointed different team members to act as liaisons to each of these teams. For example,
his software architect liaised with the two technology experts, Shane liaised with the customer, and the engineer with 22 years of experience liaised with the technological process team because he worked on the legacy system implementation. After Robert rejoined the P1 team, he remained affiliated with P2 as the liaison. Although the P1 team was struggling with technology, they never felt compelled to consult with the P2 team. To them, P2 was a different project. The P1 project manager never encouraged his team to contact the P2 team or ask for help even after it became clear that his project was lagging behind during the implementation phase. Since the P2 project manager always sent his weekly progress reports on time, the P1 project manager did not have additional needs for information about P2. The communication between the two project managers remained formal, and the P1 project manager used Robert as the messenger. It is important to mention, however, that the P1 project manager appreciated the efforts and resolve of the P2 project manager and conveyed his positive impression to the IT3 senior management.

The difference in the modes in which the two project managers and teams operated can be found in this episode. While Team B charged a flat fee for their service, Team A worked out the cost on a request by request basis. To request their services, the P1 software architect wrote a specification to which Team A offered a quotation that was never challenged. The P2 project manager and Robert did the same thing, but the P2 project manager challenged the quotation by claiming that it was too high. He said that the programming work was simple and could be completed in half the time. The Team A leader started explaining the quotation. The P2 project manager followed up by estimating the duration of each mentioned task. To that, the Team A leader responded with Why are you so concerned about this price? It’s not that much money! Robert was surprised with this verbal exchange, but then added that maybe the price was too high. Eventually, the leader agreed to lower the estimate from 16 to 10 man days. The P2 project manager concluded by saying I’m in a difficult position myself! Much later he confessed that perhaps it was not a saving worth arguing about.

There were other differences in the manner in which each project team operated. The project plan put together by the P2 team that used one week cycle to build an increment helped in developing a strong
bond. They found their work dynamic and interesting because there was always a version of the system to play with, and in the P2 office laughter could often be heard. By contrast, work by the P1 team remained individualistic, and the P1 office was quiet during our visits, with each engineer sitting silently at his desk. This difference was also influenced by the sitting arrangements. The office space for the P2 team was somewhat isolated, whereas the IT1 office could sit 80 people. The P1 team shared some understanding of their work, but their system and object design knowledge was, for the most part, developed individually by an engineer, and their knowledge of the whole system was sketchy. Most of them were not very familiar with the work of their teammates. Their knowledge and skills remained basic because their tasks were not repetitive so that they could reuse them, consider alternatives, and employ creativity with more confidence. They all contended that their work was challenging and progress was slow. Because of the work allocation and different teams involved, the decision process remained volatile, and the team structure never became recognizable.

9 IN THEIR OWN WORDS
Both senior management teams underestimated the effort, and both projects experienced difficulties, but the ways they approached the difficulties and sought resolutions were different. However, the teams were not the only party responsible for these difficulties. The customer also contributed to the slow progress with their decision to purchase the six UNIX workstations only after the systems were ready for the handover (i.e., parallel testing). Each IT purchased only one UNIX workstation because they could not justify the expense of purchasing more. IT1 estimated that they would need only one workstation to maintain the new systems, and IT3 did not have a future project in their plan at that time that would utilize the workstation. The issue of insufficient hardware resources affected the P1 team in particular because of its size. They explained to us that they tried to come up with a schedule for using the workstation, but the team was simply too big and inexperienced for a compromise solution. So they had to accept the code-compile-execute cycle to take one hour or longer. The P2 team balanced the usage of their workstation by scheduling client and server tasks in their project plan to different days.
Project 1: their own words

The P1 team never staged a demonstration for the customer or the P2 team during their implementation phase. They never resolved the issue of the new graphical user interface design until the handover process started. Neither party asked for additional meetings to finalize the new graphical user interface, nor the customer ever asked for a demonstration. The P1 project manager was confident that the new graphical user interface would not be different from the old one in its behavior, content, and layout, and the new forms would be similar. He reasoned that a demonstration would encourage operators to argue this and that, and ask for meaningless changes that would make the situation on his project more difficult. He added that after all, they wanted to keep the old system in. Another reason was that the system emerged only before the handover phase and a partial integration was never attempted. When asked about the possibility to partially integrate some of the use cases sooner rather than later, he replied that that was not in the plans. When they finally staged a demonstration of their system during the first attempt at handover, the customer flatly rejected the graphical user interface. The raw list of change requests had over 100 items. Overall, the system was unstable and required more coding and testing before it could be installed.

During the implementation phase, we interviewed the initial seven engineers in the P1 team and the software architect. When the phase started, their first task was to analyze the requirements and work on the design. They experienced many difficulties, most of which were technology related. Because the Object-oriented method was new to them, they lacked the confidence and experience to be able to change their routine in approaching and solving problems: It was easy to recognize the objects and identify their attributes, because we know about the application domain, but it is difficult to think about the behavior and how to implement it.—In the past, we simply started by writing our Fortran code because all we had to do was to add some code to the program or modify code that was already there. If a bigger change was required, we simply imitated a similar design in order to keep the program consistent. This time, we have to do everything from scratch.—We have too many new things to learn about, almost all at once. I didn't know where to start from, let alone how to combine everything together and imagine
a solution. — First I had to learn UML, and then figure out how to design and document, and then how to write programs based on my designs. I am not a beginner, but my past work was much more direct. I knew what I had to do, so I just did it. — I never worked on a multitier system before, let alone a fault-tolerant one. I am not sure if we got it right, anyway.

Other engineers made similar comments: Initially, I tried to put everything into the classes that I could identify in my use case. I couldn’t imagine what other classes I would need and what classes are available in libraries. — I didn’t know how to improve or simplify my code and design. Our training was so basic that it taught us neither. — The graphical user interface was a bit easier because we had some examples to follow. — I spent most of the time getting the syntax right. I just kept on forgetting the semicolons and mistyping names of attributes and variables. And the UNIX workstation was absolutely killing us. It was so slow that by the time I could rerun my program I forgot what the change was about. — These problems are new to us. Our previous work was much simpler. It was individualistic even when working as a team, and the team was small. — I’ve been doing my maintenance work by myself. Seldom, would I ask for advice or help. But we know each other well. We’ve been working here for years and our work was very similar. — We’ve been successful in the past. This project was a completely new experience, a completely different game. It undermines confidence. — I think the consultant lost patience with us because he was expecting different questions from us. This technology is new to us, and we keep on asking him about those simple things, so he lost interest. He just tells us where to find an example or an explanation. So, I stopped asking him questions altogether. — Inheritance is a simple concept, but how to use it to simplify our work wasn’t. It was easy to identify the common objects but it took us time to come up with a hierarchy of classes. Actually, most of our object model is flat. — I am not familiar enough with the libraries. I don’t know what classes are there to extend. — I think I’ve learnt a lot, but I ain’t confident in my knowledge and work. We have no time to experiment and improve our code.

These problems affected performance of the whole P1 team and implementation of the system: We could have more team meetings scheduled in our project plan, but perhaps we didn’t need them either because we didn’t know what exactly we should discuss and what should we all know. — It took us a while to learn enough to start talking to one
another about the system. — Most of the time, I just sit at my PC trying to figure things out. Other guys are also like that. We hardly talk to each other. It’s tough. — We wasted so much time doing the same thing over and over because we couldn’t figure things out ahead of time. We have similar code all over the system. — One person should’ve come up with a concept, a framework. Instead, we were just told what to use to get it done. — No one ever said hey, let’s see how our code works together. I guess, no one had the confidence. — The integration was a nightmare. There were misfits everywhere, attributes, method signatures, types. In the end, we were just happy to get our code working any which way possible. Nobody dared to think about improving anything. — Maybe, we should’ve planned our work better. The first guy who cracked CORBA tried to explain to us how to build an interface, but it was hard to follow what he said because I haven’t looked into CORBA. — I haven’t read anything on CORBA before; I didn’t understand the concept. I’m not a networking person, either, but I wanted to learn how it works internally. I had to go through it all by myself step-by-step before I became confident. I used up too much time on it. — It took us a while to start thinking about the shadow process, the synchronization, and such things.

The P1 team members also shared their experiences and thoughts on managerial issues: Management just would not believe us when we said that we needed more time to learn. They said that we got all the necessary training, as recommended by the technology consultants. — Management says that projects require action on real tasks and application, not learning. Our project manager became angry by repeatedly telling us that he had run out of excuses to justify the lack of progress to the customer. — Adding more engineers to the project helped reduce our work load, but it took time for them to catch up. We had to explain to them what to do, and we weren’t sure ourselves how to proceed. Needless to say, the UNIX workstation almost died under the workload. Now, there were twice as many people who had to learn something new rather than develop a system. — The learning curve was too steep, and by adding more people to the project we underwent two slowdowns in a short time. The new guys are OK, but we must explain to them what we do to make them productive as quickly as possible. We often engage in debates with them on what approach is better. — Too many things have been left to us to figure out. Our roles on the project should have been defined better, so that we all know who will
do what. We only focus on our own work most of the time. — There are too many new things to learn about. We spend most of the time thinking how to solve our individual problems. That’s what we talk about as a team, our individual problems. — Our customer tells me that the P2 team is much better than us. They are much better managed. Our manager only asks for more money, but we haven’t produced anything yet for them to see. — The [P2 project] manager does everything for his team. He writes code with them, he designs and tests the system just like all others. Our manager is a kind person, he is always first to come and last to leave the office, but I wish he could be more than just that.

We were also interested in the ongoing seminars in core competencies that had been offered by the corporation, and whether the P1 team members had participated: I got lots of training over the years as part of my personal development plan, but not much opportunity to apply it in my work. This project is the first opportunity for me to do something different and new since joining the company. — Initially, I was interested, but then I realized that I wasn’t using most of that knowledge in my work. This environment is static, the work we do haven’t changed. This project is huge for us. Everything we do now is almost completely new to us. — The training program is really good. We can learn, for example, about new managerial methods and other things all the time, but the environment is as it is. We only provide IT services to our customers, and they prefer to do projects their way. In any case, we know them and they know us. — So far I haven’t acquired marketable experience, but there are many incentives to stay with the company, such as stock options, which bring significant benefits. There’s more job security than with other companies because the corporation has been doing well and expanding its businesses and operations all the time. — Generally, the work force here is static, and there are few possibilities to advance without relocating to another city or division, but only training is not enough.

P1 was completed about 2.5 months late, and the final cost was almost triple the original estimate, due to the addition of new personnel, poor quality, and slow progress. The P1 project manager never considered a change to the requirements without first consulting with his team and then asking for additional funding. The most prominent change requests that we know off were the new algorithm and the graphical user interface, and those came at additional cost to the customer. As mentioned above,
the P1 project manager was convinced that the monitors would all have the same resolution and size. Unfortunately, the cost of P1 escalated and there was no money left to purchase the new monitors, so the customer decided to purchase only one big new monitor for each stage and install cheaper monitors throughout the facility, which were not identical. As a result, the P1 team had to reimplement the graphical user interface in order to make it scalable. Although the P1 project manager knew that the P2 system had a scalable graphical user interface, he never consulted the P2 team on the matter. The P1 team did not reuse an existing layout manager, but developed their own that would only adjust all components to the monitor size and resolution at startup. This solution simplified the change and reduced the effort. It highlights the confidence that eventually developed within the P1 team and the talent that finally emerged.

**Project 2: their own words**

Due to persistent demands by operators, the senior management decided to introduce the table into the new system, along with the histogram. The P2 project manager treated that as a minor change request, because the data were already available and it was simply a matter of displaying them in a different view and in a new form. It took Mary less than a day to implement the change. (The P1 project manager treated that as a major change and rejected the request because the customer did not want to pay for it, but later he conceded.) The P2 team gave two demonstrations of the system before the handover, and they both went well, although there were some minor changes asked for because different people attended the presentations. During the handover, there were additional minor changes asked for because all the operators were present there at the same time and had engaged in a lively discussion until they reached a compromise on the graphical user interface and how the system should ideally behave. The P2 team remained open to suggestions and they were ready to oblige. After the handover, the P2 project manager commented that [he] *would have to increase his team size by one or two engineers if the project would have lasted a month or two longer, because this team wouldn't be able to cope with the pressure. They exceeded their limits!* P2 completed on budget but three weeks late. It made almost no profit.
The other P2 team members also made a number of comments regarding their work on the project: *This is not a large system, but I find it interesting. It has many external devices to account for, and each one is a bit different.*—*We prepared well for the project. Our design is simple and reusable, which makes our coding work easier.*—*Our project plan is really good, but there's a lot of pressure to do everything on time.*—*We lost too much time at the start of the project waiting for the other project to come up with something. Now, we have to work hard to make up for it.*—*I am paid by the hour, but the company doesn't pay for overtime work. We all work long hours.*—*We work well together and we figured out a great project plan. We figured out the dependencies. Only one person couldn't do it.*—*This team is all new, but we managed to bond easily. I guess it's also the work that has been interesting. We constantly add new features and play with our system all the time.*—*It's nice to see that what we have done actually works in the system. We add a little piece every day. We all test the system all the time, but we don't have to write test code. It saves us time and we do not overlook anything.*—*We're confident. We don't expect any problems with our system. It works just fine.*—*To me software engineering is about designing; it's a creative process. In the past, I enjoyed experimenting with new ideas so as to enhance my knowledge. This project requires a different mindset and constant interaction. First, I have to figure out the easiest way to accomplish my task. Then I have to analyze my design against the system together with my colleagues.*—*We must be careful and efficient because the schedule is tight. We have to be disciplined, and think as one.*

The P2 team members also shared their experience and thoughts on their project manager: *This project manager wants to keep everything simple, but I'd like to experiment a bit. I've been reading a lot on Object-oriented method and software systems, trying to keep my knowledge up-to-date.*—*It's a new experience for me, this obsession with simplicity. The project manager says that we have neither budget nor time for experimenting. On the other hand, we experiment with the way we use technology to help ourselves to advance faster.*—*Everything we do looks so straightforward and easy. Mary reviews our code and design, and makes them consistent, as if only one person wrote it. She is strict.*—*At first, I thought that our project manager dictates everything, even when technology was concerned. Then I realized that he only wants us all to
share the same vision and think how to get things done efficiently. Then, he'd let us to incorporate our ideas into the design. — In my experience, managers use only one process model for the project. This project manager has a passion for mixing things up. We use a different model for each phase. — This is supposed to be a waterfall project. Overall, it looks like it, but we use other models, too. We overlap a lot, individually, and as a team. We build incrementally. — Our project manager is very good at finding a compromise and problem solving. We never argue about anything. — We attack problems head on. The project manager would spot it soon, anyway, and start asking questions. He gets into everything. — At times, I wish I had more input. The project manager has figured out many things for us. He asks for our opinions, but it’s his decision in the end. If he didn’t like my estimate he’d always ask me to reconsider. Then, you decide, I said to him. He’s never been assertive, though. — We work hard. The implementation gets frantic at times, but we’re winning!

10  ANALYSIS
Projects are temporary and unique organizations formed to meet goals and bring about added value or beneficial change to all stakeholders. A project is characterized by its personnel, structure, task, and technology, which managers must constantly analyze, strategize for, and take actions in order to achieve goals. The software systems development process is a knowledge-intensive socio-technical system that can produce satisfactory results only when all the needed expertise and skills are available, coordinated, and integrated (Clegg et al. 1997), such as domain knowledge and skills, and the effects of a project’s environment, job design, project management, and technology (Waterson et al. 2002). Knowledge is not absolute since it depends on a practice and a social context in which both explicit and tacit knowledge is conveyed through personal contacts (e.g., (Bresman 2010), (Brown and Duguid 1998)), and practices and strategies adopted and applied by project managers and teams vary across projects. When project managers and teams are capable and willing to approach projects and work with a desire to adapt, they can create new designs and possibilities that can make the project more likely to succeed (e.g., Kimbell 2009). The design perspective is: capable of accommodating multidisciplinary inputs; directed towards the generation of alternatives and a goal of joint optimization; explicitly
value directed; future oriented; oriented to a class of problems rather than one specific problem; and recursive, as problems are recycled as the design develops (Cherns 1973). In our analysis below, we find that the P2 project manager made use of these, whereas the P1 project manager did not, which, in our opinion, made their respective projects successful and unsuccessful—although both projects exceeded their cost and time estimates.

We begin this analysis by addressing the 10th success criterion in Table 1. The exercise of hard work and focus proved insufficient for this program to succeed. Both teams performed unevenly, but proved hardworking and determined to deliver a quality product to the customer, although their sources of difficulties, expectations for the outcome, and motivation for the effort were all different. The P1 team was saddled with a reactor strategy (Miles and Snow 1978), due to being low on exploitation and exploration, The P1 team could neither anticipate and plan for the future nor assess the present, and problems emerged as surprises for which there was not enough competence (i.e., abilities, knowledge, and skills (Lee et al. 1995)) and managerial savvy and willingness to deal with proactively. The P1 team first needed to acquire and develop solution domain competence, which became a challenge. The training was insufficient, and the lack of competence at both the engineering and the managerial level created other issues, such as the lack of a compelling vision, desire to adapt, and a fitting strategy; overstaffing; and poor collaboration and communication within the team. Yet, the P1 team did not lose their resolve to succeed because of their desire to broaden their competencies and their familiarity with the environment, whereas the P2 team developed the resolve because of the path the P2 project manager chartered for his team. He designed his project to succeed by opting for a defender strategy, i.e., by deliberately trading exploration for exploitation. The P2 designed simple solutions and executed tasks in a manner that geared the use of resources towards efficiency. The team focused on cost, proactive risk avoidance, quality, and results, and the P2 project manager engaged in a candid and receptive dialogue with the customer.

The P2 team was new to the environment, but to them and to IT3 this was just another software development project to work on. The P2 engineers lacked interest in the application domain, and they rarely
engaged outside the team because their peers at IT1 lacked solution domain competence. The P2 project manager responded by taking remedies that were enabled by the team size and justified by the tight budget. After a slow start, the P2 team managed to bond and work well together, being motivated by positive developments on the project, and leadership by the P2 project manager who was determined to lead a successful project in spite of difficulties and the epistemic divisions and politics between IT1 and IT3. It is likely that politics motivated the P1 team never to initiate a supplementary knowledge transfer between the two project teams, and the customer never to sympathize with the project teams that were both affected by insufficient hardware resources. Researchers have found that team leaders should engage in many different kinds of behaviors intended to foster team effectiveness, including arranging for the resources a team needs for its work and removing organizational roadblocks that impede the work, helping individual members strengthen their personal contributions to the team, structuring the team and establishing its purposes, and working with the team as a whole to help members use their collective resources well in pursuing team purposes (Hackman and Wageman 2005). However, in Table 1 the impact of leadership on project performance has not been ranked at all, and the competence criterion was ranked low although, on this program, it affected managerial decisions, performance expectations, and the teams’ behavior.

The performance expectations for P1 were not realistic, because the IT1 senior management underestimated the needs for training both in breadth and depth. It proved naïve to expect that the basic training sessions could prepare the P1 team for the paradigm shift such that they could undertake their work with the confidence and ease they were used to. Research has shown that a method learned from a crash course or a textbook is essentially based on topic knowledge, whereas software development requires episodic and topic knowledge (Robillard 1999). A learner experiences proactive interference when the existing knowledge that is now out of date interferes with the learning process because it is inappropriate in the new domain and, hence, a source of confusion and errors (Bjork 2011). The P1 team also lacked competence in software systems, for which they received no training at all. All these were not affected by user involvement, which explains the success of
the P2 team. In the environment where requirements were clear and stable, and application domain knowledge was readily available upon request, the P2 team had a clear advantage given their solid project management and solution domain competence. Although the IT3 senior management underestimated the effort required because the application domain was new to them, it did not make P2 unprofitable due to the goals and vision developed by the P2 project manager of a dependable system with a simple design within the shortest possible time, as well as his assessment of the solution domain competence of each P2 team member to which the project was adjusted. By contrast, the IT1 senior management expected that the competence of their personnel would be on a par with the challenge, but their vision of the new system, if any, certainly did not resemble the P2 project manager’s vision.

On this program, application domain knowledge played a lesser role than solution domain competence in two important aspects. Firstly, in this environment, application domain knowledge was available from many sources, and on P1 there was no shortage of it, including the project manager. The commanding application domain knowledge gave the P1 team the false impression that they knew all the answers. Customer participation was neither desirable nor necessary, because the P1 project manager perceived it as the main source of risk. He kept input by the customer to a minimum, justifying the strategy with his past experience when change requests bearing little significance undermined the project. These can explain why he was not concerned about the unfinished graphical user interface specification, and why the P1 team never felt compelled to stage a demonstration of the new system to the customer before the implementation was completed. The P2 project manager, by contrast, was open to customer’s input because the customer would decide whether to accept or reject the system. To him, working with a customer was unconditional. As a result of the opposite attitudes towards the customer, the level of customer involvement on the two projects became a matter of managerial preference. Secondly, P1 failed to make up for the lack of solution domain competence. At the start of the project, neither the P1 project manager nor his team knew how to design a project and a system for the new technology that would be tailored to their level of solution domain competence. The involvement of the consultants and experts was not beneficial until the team acquired
enough knowledge to integrate that expertise (e.g., Tiwana and McLean 2005), and it remained informal and limited. Therefore, P1 had nobody who could anticipate problems and shortcut their way out, let alone develop a design perspective, and the level of their competence affected the quality and span of their engagement.

Both project managers tried to avoid risk, but from a completely different viewpoint, which also revealed their different backgrounds, experience, and managerial preferences. For example, the P1 project manager’s preference for delegation was high, but his ability to integrate and unify was low, whereas the P2 project manager preferred moderate delegation and only when the goal was clearly established and its overall implications were understood by his team. Projects are exposed to risks driven by many factors originating from their changing and unique climate and environment, and proper planning and smaller project milestones are important success criteria (see Table 1). The roles of engineers and managers do not exist in isolation, and this awareness should be embodied by all team members, so as to share the mental model and internal ownership of the project. Mental models are a means by which individuals and organizations create and share meaning, thereby enabling a common understanding and the development of knowledge (Hill and Levenhagen 1995). The mental model of each individual can prove decisive for a project’s outcome, and even more so of a project manager. What many organizations call planning is simply a projection of their current mental models into the future (Arrango 1998), and [many] managers operate on mental representations of the environment that are likely to be of historical environments rather than of current ones, being first committed to policies and procedures (Kiesler and Sproull 1982). Our empirical study has taught us that, in order to manage a successful project, a project manager’s or a project management team’s competence should be broad and possess an all-inclusive attention to details. A project management team must assess the climate and environment, develop a new mental model, strategy, and vision, and design a project accordingly. The project manager must communicate these through various means back to the environment in order to gain and maintain the support of all stakeholders for the project.

In order to create a project plan, both project managers in this empirical study engaged the project team in the planning process, but the
engagement was not reciprocal. The P2 project manager made estimates together with his team and helped them with other work, whereas the P1 project manager could do neither. The P1 project manager was not familiar with the personnel and technology to design the project to facilitate the task. He neither asked for nor received any training that could help him manage this project. Instead, IT1 assumed that his past experience would suffice because this was the same application domain and environment and he was their most experienced project manager. The P1 project manager and his team did not develop a new mental model that would be appropriate for the project. The P1 software architect was new to the technology just as the team was, but they did not explicitly qualify this as a risk, even though they overestimated tasks to account for the learning needs. The P1 team assumed that the project was doable because they implemented the legacy system and the new technology was mature. IT1 believed that they provided ample external and internal support and good structure for both projects to succeed. In contrast to these, the P2 project manager initially found himself in an unenviable situation in which, besides being a newcomer, he had little control over and insight into his own project. The P2 project manager used his managerial and technology background and work ethics to turn the situation around. He identified multiple sources of risk, such as budgetary and scheduling constraints, personnel competence and their lack of motivation to acquire application domain knowledge, task complexity, and technology and its application. The P2 team utilized multiple risk management techniques, such as design for dependability, flexibility, maintainability, and to competence; dynamic process tailoring; prototyping; and ongoing system testing for the progress and quality monitoring. The P2 team worked diligently on the analysis and design so that they could define frequent milestones and smaller tasks. The P2 project manager was leading, whereas the P1 project manager was only observing his team’s effort.

Nevertheless, both project managers were determined to make the projects successful, but their individual contribution, influence, and involvement throughout the projects were different. The roles of the project manager and the software architect were assigned to different individuals on P1, which is common in the practice. But the separation raised here questions of influence and responsibility in those areas of
project planning and execution that required input from them both. For example, the P1 software architect did not think that his responsibilities included assessing the competencies of team members and reconciling them with the needs of the project, or enforcing that individual decisions meet a design goal. The P1 project manager never engaged in technology-related issues so as to exercise his own authority or influence, or form his own action plan or opinion. The lack of governance extended beyond the project team. The software technology experts were formally a high level technology management team who collaborated with the P1 software architect and the TCF team. They decided on the three-tier architecture for the new systems, selected the technology, and together with the TCF team designed the infrastructure and co-supervised its development. However, we learned that the software technology experts were not directly engaged in the other issues that affected P1. Consequently, the P1 team approached each use case independently, thereby creating an isomorphic team structure. Because there was no integrator or mentor among the project leaders, the design decisions and work allocations followed individual preferences. In his project plan, the P1 project manager assumed that each activity to implement a use case consisted of three tasks, i.e., a graphical user interface, a server, and a database task, and the estimates were made by those responsible for the use case implementation. Unfortunately, when tasks are not simple and well-defined, it is not easy to identify and understand which parts of the task affect the other parts (Espinosa et al. 2007), to learn, and to share.

Research has shown that a multi-stakeholder project must carefully position itself to its environment, and the goals and management methods of the project must be carefully matched with the context and the situation at hand (Karlos et al. 2008). Yet, the strategy adopted by the P1 project manager included a static project plan, and the waterfall model as suggested by his customer, both of which he grew accustomed to in the past. During the requirements elicitation phase, the P1 project manager engaged only those who worked on the task. The others were free to use their time to their liking or help with urgent tasks on other projects. Since they were not contractors but regular employees, their formal affiliation with P1 did not affect the budget because they were not doing billable work. Only the two novices had to acquire the application domain knowledge. This explains why Robert was available...
to work fulltime on P2 requirements. The lack of urgency and targeted preparatory work for the implementation phase further weakened the performance of individuals and the P1 team. Even at the end of the project, the P1 team members could not explain to us much about the new system except for their own individual work and the difficulties they had experienced. All these factors led to poor control over the project, and limited their capability to deal with problems to the recruiting of more engineers to the team.

Both projects used the same quality procedures, such as change control, coding standards, formal control and planning tools, and formal reviews (e.g., McFarlan 1981). They used external and internal integration tools, such as a steering committee, formal approval process, and regular meetings and progress reports. The P1 team was fully accustomed to them, but the P2 team was not, and the external and internal environments of P2 were different. The P2 project manager faced three external sources of difficulties: the different business environments at IT1 and IT3, the unfamiliar application domain, and the politics between IT1 and IT3 regarding P2 cost and future distribution of business that was boosted by the lack of a governance structure. The P2 team was operating on its own, without the mandatory support from the other teams, and did not participate in the training for the P1 team, because IT3 assumed that they were all well versed in the technology. There was no application domain training because neither the customer nor IT1 would provide it. The customer knew that IT1 was familiar with their business and technological processes, and the outsourcing of P2 to IT3 was not their concern, which also explains why P2 lacked formal links with the customer. IT1 did not want to train IT3 personnel, arguing that the problem would be sorted out through reuse of their code and design, and IT3 treated P2 as a software problem. Internally, the P2 team had to be built because its members were unfamiliar, whereas the P1 team had strong ties, and knowing one’s teammates facilitates team performance (Harrison et al. 2003). IT1 and the customer had strong ties, whereas the P2 project manager and sponsor had to learn how to work together and with the customer. The P2 team building process was further hampered by the distance from the customer and IT1, which forced the P2 project manager to spend most of his time away from the team during his first month on the project.
The P2 project manager acquired his project management experience at various companies, and his technical background by working on a number of challenging software development projects. He was the only member of the P2 team without fundamental unknowns in the solution domain. He understood the concepts and the foundations behind the technology, and he knew how to use it for the benefit of the project. He used his broad competence in various ways: during his contacts with the customer he discussed and estimated change requests with confidence, and also helped in overcoming the customer’s lack of computer literacy; during the project planning he discussed estimates and identified task dependencies with his team, matched designs to skills and simplified tasks, and assigned roles and tasks so as to engage the team on broad issues; during the implementation phase he engaged in the solving of coding and design issues, and, finally, he motivated his team by demonstrating and explaining to them an easy way out, and led them towards the goal. Because he acted both as project manager and software architect, his plan for the project and the system encompassed the assessments of the development environment, the individual competences of his team members, and the tasks. He designed the project so as to minimize the exposure to such risks and was constantly looking for solutions that could improve the performance of his team. For example, Mary and Corey wanted to experiment with new ideas, but the P2 project manager repeatedly cautioned his team that their customer only wanted a dependable and fully functional system that was delivered on budget and time and not a state-of-the-art software design.

Both P1 and P2 were waterfall projects, but the P2 project manager tailored the process model as the environment and phases changed. By doing so, he controlled the cost and risk, improved the efficiency by overlapping activities, as well as utilizing incremental development, repetitive work, and simple architectural components that could be reused without modification, etc. His application domain knowledge was only emerging, but he did not hesitate to undertake a side project because that could be a good exercise and test for his team that could further boost the emerging positive image of IT3 with the customer... based on mutual respect and trust. During the requirements elicitation phase, the P2 project manager engaged his team on the analysis and design based on those P1 requirements that were completed and relevant to
P2. The P1 project manager did not have that opportunity, but his lack of urgency to engage the team early on in project related matters was, nevertheless, puzzling to us.

The P2 implementation phase consisted of short tasks and small increments, and most of the milestones took a week to reach because the schedule was tight. The architecture first strategy made work of the P2 team easier and interesting, and it facilitated quality because there was always a system to play with and test as they were implementing one use case at a time. By contrast, the software architecture on P1 only emerged at the end of the implementation phase, and the system remained riddled with problems and was unstable well into the testing phase. The P2 schedule included reciprocal tasks, but these tasks were mostly small and well defined, which made mutual adjustments easy to achieve, and gave the team the ability to integrate code into the system and test it without a delay. The P2 team was collocated and small, and it was not a problem to have tasks that required frequent communication and coordination. The P2 project manager demonstrated how to use managerial competence to tailor a project plan and strategy over time by nesting different process models and by instantly incorporating the learned lessons, instead of using only one strategy (i.e., the waterfall), as the P1 project manager had done. The P2 project manager also demonstrated how to use technology for the benefit of his team rather than letting it dominate the project. He took the internal project ownership upon himself, whereas on P1 it was never clear who was supposed to make project level decisions.

On the negative side, although the P2 project manager insisted that quality was everyone’s business including his, the P2 team lacked interest to delve deeper into the application domain and interact with the P1 team, for which they saw no value for their careers. They approached their work strictly as a software problem. When they did not understand something, they passed it on to the P2 project manager to provide the answer. Since the project was small enough for one person to grasp all of its aspects, being the liaison allowed the P2 project manager to gain full control of the project. Even so, all the P2 team members were familiar with the system because their project plan with short tasks and frequent milestones had teamwork in mind, i.e., it forced them to remain engaged, interact through their work products, and thus get to know the system.
and share expertise and ideas. The P2 team had no significant problems with the application domain, and the system handover and live testing went smoothly, which spoke to the clarity, quality, and stability of the requirements and the product. Given all these and because the lack of a vision negatively affected P1, we can conclude that the criterion of a clear vision proved more important than as suggested in Table 1.

The user involvement was important although, at times, their conduct and decisions could be questioned, such as the customer’s decision not to offer their hardware to speed up the implementation. This became even more puzzling after the hardware was provided for the side project; however, it was explained to us that TD purchased the personal computer and the board to connect a thermal probe to it. The goals (i.e., objectives in Table 1) and the scope were clear, but they were stable only for P1, whereas for P2 they changed as they learned about the environment and project, and after the projects split up. The proper planning and the smaller project milestones criteria were important on P2, whereas the P1 project manager lost control over his project and this was due to the unrealistic expectations regarding himself and his personnel. For the P1 team the learning curve during the implementation phase was steep, which affected their teamwork. The P2 team also experienced teamwork issues, both horizontally and vertically, which was due to teambuilding. Both projects enjoyed full senior management support, which was negatively affected by the governance issues.

The IT1 director was in rapport with the P1 project manager and, to the best of our knowledge, so did all teams at IT1 (see Figure 2). But the lack of engagement suggested that some roles were narrowly defined. The lack of clearly defined goals and roles and its negative effect on the performance of the P1 team persisted. On P2, the communication and the decision-making process between the project manager and the project sponsor took time to develop, which created intrateam issues. At times, the P2 sponsor expected feedback from the P2 project manager without asking for it, whereas the manager expected more credit and support from the sponsor. After a P2 governance structure was established, the tensions subsided, but the interteam governance issues persisted because IT3 neither officially announced who was the person in charge of P2 nor who was the P2 contact person. The IT1 director and the P1 project manager assumed that the P2 project sponsor was
the person in charge of P2 because *that was where the buck stopped*. For example, when the P2 project manager contacted the IT1 director via email he sent a reply to the P2 project sponsor, as if he had asked the question. These kept the P2 project manager out of the information loop and created unnecessary and unpleasant surprises.

It is worth establishing how the program affected the collective and individual learning in order to perform better the next time. As one can conclude from the feedback that we collected in our interviews, the P1 team members were not convinced that the newly acquired competence had sufficiently developed to make them confident that their future assignments would fare significantly better. To them, the project was too difficult to consider alternatives and contemplate improvements. By contrast, some P2 team members were disappointed that they could not be more creative, but they appreciated that they managed to complete the project on budget. They argued that *they would not have missed the deadline if they could have started the implementation as originally planned.*

It was interesting to notice that the P1 project manager appointed Robert as the new customer liaison instead of Shane. When asked to comment on his decision, the P1 project manager replied that *Robert had more experience*, but it was difficult to conclude whether the whole episode points to single loop learning (Argyris and Schön 1978), or is merely an unreflective action taking (Fiol and Lyles 1985). Nevertheless, both Robert and Shane had arguably done an excellent job on requirements, just as they had spent years working well for the same customer. To the best of our knowledge, the customer senior management did not perceive the problems that burdened P1 as Shane’s mistakes. They did not attribute the graphical user interface problems to him, just as they did not attribute the accomplishments of the P2 team with their graphical user interface to Robert, because he neither suggested to build a prototype, nor got involved in any of the subsequent change requests. The customer was aware that those decisions were influenced or made by the respective project managers, and they noticed the difference in the approach and conduct that they experienced while working with the two.

Finally, we describe the impact that the two projects had on IT1 and IT3. Upon implementing P2, IT3 took advantage of the fact that IT1 was still busy working on P1 and engaged in another project for another in-house customer that had also been processing raw materials.
That customer was located in another province, but IT1 maintained the system. The project was nearly identical to P1 because the customer had only one stage, which was similar to Stage 1, but its processing capacity was smaller. The customer appointed a new project manager (NPM) who had an office at IT1, and he consulted the IT1 management for advice. The P2 project manager initiated the negotiations, and together with the P2 project sponsor negotiated the terms with the NPM. This became a greenfield project, and the NPM asked IT3 to allocate eight engineers to the project, with IT3 being happy to oblige. The P2 team was assigned to the new project with the exception of the P2 project manager, who remained with P2 until the system handover was completed and his contract with IT3 expired. The NPM rehired the TCF team leader who became the technical leader for the project and the team leader in charge of the eight engineers at IT3. After completing P1, the P1 project manager became responsible for the development of a new system for Stage 3. The Stage 3 project was fixed priced, and for which the P1 team shrank down to six engineers. The P1 project manager decided to move from the cubicle back to his office, and the P1 software architect decided to rejoin IT2.

11 CONCLUSIONS
The literature concerning software development projects has mostly focused on a specific aspect, and its presentation has been weak on details, particularly in quantitative empirical studies. For example, the key constructs derived from project management literature were characterized by such terms as inadequate project milestones, poor control, poor customer relations and specification, properly planned, and underestimation of resources. All, any, or a combination of some of these can affect software projects; but because they require a context, we wanted to address two issues that remained vague: a) how or why something happened, and b) what was or was not done to anticipate, identify and resolve issues, so that we could connect the cause and effect with the stakeholders and environment. For example, project size is a complex factor. In this study, the size estimates were often based on the insufficient understanding of issues or needs, which extended to other decisions. In the analysis, we verified whether our findings were backed up by relevant research and theories, and if the knowledge to
avoid or overcome the issues was available. By doing so, we assisted the corporation in improving their organizational learning program.

In the domain of work, the model of technical work and techniques has monopolized discussion in the business and organizational literature since Taylor, at the expense of our understanding of practice (Orr 1998). The best choice is one that jointly optimizes both the social and the technical systems (Cherns 1973). On these two reengineering projects, we found that the managerial and engineering roles could not be separated. It was hard to manage the systems project with little or no understanding of how to design the jobs, the structure, and the system, and use the new technology to fit the project and meet the needs of all stakeholders. One team was familiar with the customer and knew the legacy systems inside out, and had external and some important internal integration tools, all of which the other team lacked. The familiar team finished the project significantly behind schedule and over budget, and the other was only late. Instead of a single criterion in Table 1, a combination of criteria proved decisive. Both projects met criteria 1, 2, 3, 8 and 10 (59% project success ratio). Both teams missed criterion 7. They diverged on criteria 4, 6, and 9 (0 or 23% project success ratio). An important finding is that Criterion 5 remained inconclusive, because different time or view points gave different answers that were all related. For example, a poor assessment led to a poor decision or unrealistic estimate; the insufficient training was a team performance obstacle; and the size difference between the two projects and its evolution rendered the managerial expectations relative.

We find this evaluation credible and relevant because the projects developed nearly identical systems in a natural setting, and used the same methods and tools. We also find this evaluation inconclusive, as Table 1 missed criteria such as mismanaged decisions and issues within a team. An important finding of this study is that the strengths and weaknesses applied equally to the engineers and to the managers. The stakeholders sometimes acted counterintuitively to an observer, because the governance structure either took time to develop or was missing. The individual competence and experience of the managers became a major factor in the outcome, such that, from the very start, the two project managers were in charge of two different projects, but for different reasons. The initial plan for reuse to cut cost became infeasible and
the projects diverged, with each reflecting the background, engagement, influence, and mental model of the manager. While the familiar project manager primarily reacted to developments on the project and avoided change, the other project manager found change unavoidable and tried to act proactively. The individual competence and experience shaped the ability to develop a comprehensive project plan, the anticipation and perception of events, and the involvement in the process of seeking resolutions to challenges and risks experienced on the project. Change on a project can come from many sources, and professionals should possess the ability to cope with changing and complex environments (Bassellier and Benbasat 2004). If change does not necessarily imply learning, adopting new technologies certainly requires it. Project managers should possess competence in dealing with nonroutine problems and tasks and in finding ways of linking business and technology goals. In this evaluation, the success of the other project gained significance.

We found that competence, objectives, and vision deserve much more attention than is suggested in Table 1. The project with more solution domain competence had a clear advantage in criteria 4, 5, 6, 7, 9, and 10, which commands further research at all organizational levels and from multiple viewpoints. We found that creativity, decision making, and experimentation required competence, control, and direction (i.e., criteria 4, 6, 7, and 9) in order to become beneficial for all stakeholders. Organizational adaptation and learning require a concerted effort, and a proper environment and strategies. The failure or success of a project becomes even more relative when it deals with a domain or technology that is of strategic importance for the company. If the personnel neither develops nor feels confident about competence in either of the domains, then the company has not fully profited, irrespective of what the outcome was. Software systems development is much more than programming. It is a systems problem and scattered interventions cannot significantly enhance overall project performance and the quality of the final product. Some of the techniques used by the other project were well known, such as the incremental development, and the ongoing testing for control and quality. The overlapping of phases was used for better resource utilization and schedule compression. Novel ideas emerged that require attention in our future work, such as strategizing for the purpose of dealing with personnel, task, and technology risks by the use of software
architecting and designing, overlapping projects for cost cutting, nesting of process models, and dynamic process tailoring.

Our assessment of the program, according to Table 1, put the prospect of successful outcome at 60%, yet it missed expectations. Only by examining the four interdependent variables that characterize socio-technical systems (i.e., personnel, structure, task, and technology) can one appreciate the curious and surprising outcome of the program. Managers can control and plan for these four variables only when they are familiar with all of them. We do not suggest that the project manager be responsible for all four variables, but the project management team must be. In this study both scenarios could be studied, and the other project manager became the difference-maker in the performance of the two projects. The two systems were relatively modest in size and in their use of software technology, but were nevertheless complex due to their environment and requirements. Size and technology became important because a person with both engineering and managerial competence could grasp the whole environment and project, catch up with the new knowledge areas, and develop a plan and vision of a successful project. In the end, both systems were handed over to the customer, who was neither satisfied with the cost nor lead time. The outcome and aftermath of the projects also revealed the breadth and depth of collective and individual involvement, knowledge, and learning at all the levels, as well as the management concerns and practices that did not favor the customer.

REFERENCES


Cherns, A. B. The principles of sociotechnical design, Human Relations, 29(8), 1976, pp. 783–792.


Kofman, F., and Senge, P. M. Communities of commitment: The heart of learning organizations, Organization Dynamics, 22(3), 1993, pp. 5–23.


Weick, K. E. Educational organizations as loosely coupled systems, Administrative Science Quarterly, 21(1), 1976, pp. 1–19.


Questions

<table>
<thead>
<tr>
<th>No</th>
<th>Question</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How did the P2 team test the system?</td>
<td>• Was quality the only objective? What other benefits and considerations were taken into account?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Did testing affect the project insight?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How would you argue against or for this strategy as a project manager and as an engineer?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compare this strategy with some other you are familiar with, such as the test-first programming concepts?</td>
</tr>
<tr>
<td>2</td>
<td>What mechanisms were put in place to motivate employees to advance their competencies and stay with the corporation?</td>
<td>• Describe the model at different organizational levels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Do you think it provided expected benefits to the employees and the corporation?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Is it better to outsource training or use internal instructors?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How would you define the core competence at each organizational level mentioned in the study?</td>
</tr>
<tr>
<td>3</td>
<td>As a project manager, how would you avoid or overcome the difficulties experienced by the P1 team?</td>
<td>• What competencies would you need?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• What strategy would you use?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How would you get the other IT1 teams to more productively contribute to the P1 team?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• How would you know that your peers and subordinates are on par with the challenge?</td>
</tr>
</tbody>
</table>

Exercises

Exercise 1: Employee Competence

For an organization to become successful its employees should possess the competencies that match the challenge. How would you ensure, through recruitment, selection, and training that employees can indeed perform as expected? If you were the IT1 director, how would you go about preparing and staffing the P1 team? If you were the P2 sponsor, how would you negotiate P2? What would you need to know to ask the right questions or solicit the right advice?
Exercise 2: Project Environment

How would you improve the contributions and relationships between the teams singled out in the empirical study? How would you engage the teams to achieve the goals, namely cost, quality, reuse, and schedule? Would you consider a different strategy instead of having two (independent) projects? What managerial and/or other competence would you need to have to implement your ideas?

Comments

You can consult the articles listed below under Further Reading before working on the exercises.

Further Reading

The following paper investigates the problem of organizational learning and software systems development over the course of multiple concurrent and successive projects within the same division of a telecommunications corporation.


In this chapter, the authors also tackled problems that were related to organizational practices and processes. The following two articles are recommended not to promote lean, but as an attempt to paint a broader picture if not to look at the totality of the problem, both of which has been seldom the case. The articles provide another take at how systems approach and thinking can change and shape the corporation and industry, and facilitate transfer of knowledge between industries. These articles testify that problems that burden the software systems development are common across engineering disciplines and industries, and the manager could have even more challenges to deal with and overcome. Some have been more successful at tackling problems while others have been lagging behind despite advances.

