Information and Software Technology 52 (2010) 680–696

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Empirical validation of the Classic Change Curve on a software technology change project

Uolevi Nikula [a,*], Christian Jurvanen [b], Orlena Gotel [c], Donald C Gause [d]

[a] Department of Information Technology, Lappeenranta University of Technology, P.O. Box 20, FI-53851 Lappeenranta, Finland
[b] Agricultural Data Processing Centre Ltd., P.O. Box 25, FI-01301 Vantaa, Finland
[c] Independent Researcher, New York City, NY 10014, USA
[d] Thomas J. Watson School of Engineering and Applied Science, Binghamton University, P.O. Box 6000, Binghamton, NY 13902-6000, USA

ARTICLE INFO

Article history:
Received 30 October 2009
Received in revised form 5 February 2010
Accepted 9 February 2010
Available online 12 February 2010

Keywords:
Organisational change
Technology change
Classic Change Curve
Technology S-curve
Change resistance
Organisational performance

ABSTRACT

Context: New processes, tools, and practices are being introduced into software companies at an increasing rate. With each new advance in technology, software managers need to consider not only whether it is time to change the technologies currently used, but also whether an evolutionary change is sufficient or a revolutionary change is required.

Objective: In this paper, we approach this dilemma from the organizational and technology research points of view to see whether they can help software companies in initiating and managing technology change. In particular, we explore the fit of the technology S-curve, the Classic Change Curve, and a technological change framework to a software technology change project and examine the insights that such frameworks can bring.

Method: The descriptive case study described in this paper summarizes a software technology change project in which a 30-year old legacy information system running on a mainframe was replaced by a network server system at the same time as the individual-centric development practices were replaced with organization-centric ones. The study is based on a review of the company's annual reports, in conjunction with other archival documents, five interviews and collaboration with a key stakeholder in the company.

Results: Analyses of the collected data suggest that software technology change follows the general change research findings as characterized by the technology S-curve and the Classic Change Curve. Further, that such frameworks present critical questions for management to address when embarking on and then running such projects.

Conclusions: We describe how understanding why a software technology change project is started, the way in which it unfolds, and how different factors affect it, are essential tools for project leaders in preparing for change projects and for keeping them under control. Moreover, we show how it is equally important to understand how software technology change can work as a catalyst in revitalizing a stagnated organization, facilitating other changes and thereby helping an organization to redefine its role in the marketplace.

1. Introduction

Moore's Law predicts that the number of transistors on a chip will double about every 2 years [36]. The resulting improvement in computer performance has contributed to many changes in information systems, development tools, and practices, making software development managers wonder whether it is time to change their existing technology solution with every new advance that occurs. In the face of continuous technology change, decision makers are consequently faced with the age-old dilemma of either continuing to refine their existing technology at the evolutionary level or with adopting an entirely new technology at the revolutionary level. The former strategy introduces minimal organizational change and, as a consequence, appears to be an immediate and low risk approach. The latter strategy, by contrast, has the potential to introduce large changes into an existing organization and thus incurs more immediate risks, but it can potentially attain higher benefits in the longer-term.

Human response to change has been a formal topic of interest since the Second World War, when the need to send men to the battlefield required changes at home as women entered the workplace [26]. The seminal studies from this period defined change as a simplistic process consisting of unfreezing, moving, and freezing steps [44], and began to identify common change resistance

* Corresponding author. Tel.: +358 5 621 2839; fax: +358 5 621 2899.
E-mail addresses: uolevi.nikula@lut.fi (U. Nikula), christian.jurvanen@mloy.fi (C. Jurvanen), olly@gotel.net (O. Gotel), dgause@stny.rr.com (D.C Gause).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.02.004

indicators in the workplace, including grievances, high turnover rates, low efficiency levels, and restriction of output [19]. More recently, the role of change has attracted increasing interest in software development circles and it has been studied, for example, from the change management [3,43,46] and motivational [63] viewpoints. However, in the software process improvement context, change has not been studied as extensively to date. Stelzer and Mellis [71] note that in their literature study of success factors of organizational change in software process improvement, unfreezing the organization was mentioned only in 24% of the ISO cases and in 52% of the CMM cases, quality initiatives from the International Standards Organization and Capability Maturity Model process framework respectively. Allison and Merali [2], on the other hand, report a structurational analysis of process improvement in a software package organization over a 10-year period focusing on the contextual and social factors of the changes.

Given its intent to characterize the general change process, our goal in this present study is to validate the Classic Change Curve [66] empirically on a software technology change project. In particular, we are interested in exploring the reasons for initiating software technology change, understanding how this kind of change project unfolds, and determining those factors that affect the project unfolding. Since the present change project started as a technology change project but ended up dealing with organizational changes, we first present the technology and organizational change research frameworks used in the study (Section 2). The actual study started by reviewing the annual reports of the company, and the questions raised were further discussed in five interviews with company employees as well as with a key stakeholder of the company (Section 3). The case study approaches the research questions from the point of view of a legacy information system redevelopment project in which a 30-year old mainframe system was replaced with a network server system and where systematic software development practices were concurrently introduced into the company concerned (Section 4). We cover both the software technology change project and the sustained software process improvement phase that occurred thereafter to see how the company evolved over a 10-year period and emerged from the "Death Valley" of Change [26]. The empirical treatment of the research questions is followed by a discussion of each question based on related research findings (Section 5). We close the paper by exploring the implications of this case study and the generalizability of our findings (Section 6).

2. Related research

The related research focuses on defining the key concepts and context of the present study by studying what leads to a technology change, the manner in which a typical change project progresses, and how technology changes can be characterized. The technology S-curve, the Classic Change Curve, and the technological change framework provide simple communication tools for discussing these complex topics. For the purposes of this paper, we define technology in its broadest sense as the processes, tools, and practices that are used in software development.

2.1. Why change a technology?

The need to change a technology becomes apparent when the existing technology reaches a natural limit. Development efforts when undertaking a technology change tend to progress slowly in the beginning but, after all the essential knowledge has been learned, subsequent progress can be quick (Fig. 1). However, at some point in time the improvement rate slows down, making incremental improvements more difficult and more expensive.

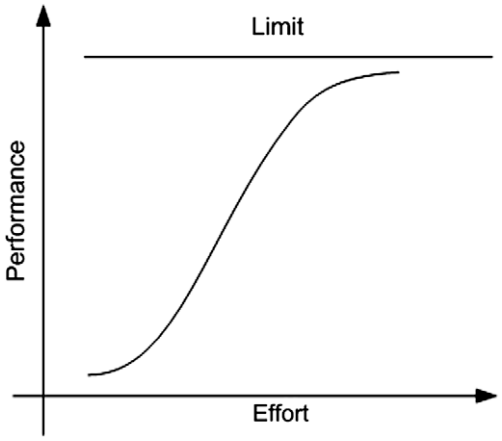

Fig. 1. The technology S-curve [27].

For example, the number of transistors that an integrated circuit can contain has a physical limit defined by the size of the components, the line width, the size of the chip [27], and the temperature buildup due to chip density. In software development, such physical limits are rare, but legacy information systems have been reported to exhibit aging symptoms such as increasing maintenance costs, limited performance, integration problems, restricted extension possibilities, and little availability of qualified maintenance and development personnel [1,8–10,69]. Since the aging symptoms, individually or jointly, can become the reason to change the system, they are potential limits of the technology currently in existence.

Another reason for a technology change is competition. Competitors using the same technologies should not pose an unmanageable threat, but disruptive innovations can provide entrant companies with an attacker's advantage [27] in a marketplace. A disruptive innovation introduces a new kind of product or service that mainstream customers seldom find interesting, often due to an inferior performance by traditional performance metrics [17]. However, since disruptive innovations are frequently less expensive than mainstream products, they can create new markets and attract mainstream customers as they mature. Examples of disruptive innovations include the introduction of personal computers, which has been claimed as the reason for Digital Equipment Corporation's "abrupt fall from grace" [17] and for none of the independent disk-drive companies of 1976 existing in 1995 [13]. As many incumbent companies have lost their leading market position to entrant companies due to disruptive innovations [13], keeping a close eye on emerging technologies is a key issue for company research and development. This also places an added burden of sponsoring active competitive analysis and market needs determination within those companies competing in rapidly changing technology markets.

A change from one technology to another, where a new technology is studied and adapted in a company context while an old technology remains in production use, typically results in a discontinuity and redundancy (Fig. 2). It has been estimated that leadership changes hands in approximately seven out of ten cases when such discontinuities strike [27] and that two thirds of major technological changes in organizations fail, mainly due to change resistance [47].

2.2. Personal and organizational response to change

Systematic study of change since the late 1940s [44] has resulted in numerous change models. Both personal and organizational level change models exist that explain, for example, the characteristic nature of personal grief [42], workplace morale

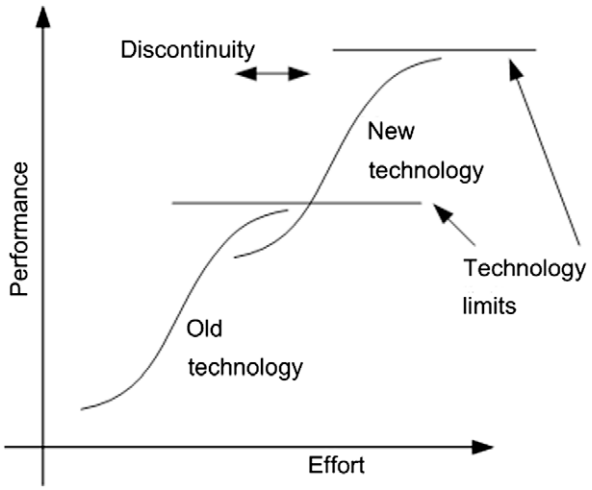

Fig. 2. Technology change leads to a discontinuity (adapted from [27]).

Table 1
Different forms of resistance and the way in which they are manifested [47], as summarized in [45].

Form of resistance	Manifestation
Confusion	Difficulty in realizing that change is going to happen
Immediate criticism	Rejecting change before hearing the details
Denial	Refusing to accept that things have changed
Malicious compliance	Smiling and seeming to go along, only to demonstrate a lack of compliance later on
Sabotage	Taking actions to inhibit or kill the change
Easy agreement	Agreeing with little resistance, without realizing what is being agreed to
Deflection	Changing the subject and hoping that it will go away
Silence	Complete absence of input, which may be the most difficult resistance to deal with

[48], and organizational performance [66]. While quite distinct in their details, all the proposed change models have something in common: they all depict a transition period falling between two states of initial and final equilibrium [26], a general pattern that was first suggested in the seminal studies of Lewin [44]. Change resistance has been observed to manifest itself in individual behavior in numerous ways, either consciously or unconsciously [47], as summarized in Table 1. Change resistance emerges from the fact that every change also means a loss for somebody, such as an expert's role in an organization [32].

Organizational performance changes have been described pictorially using the "Classic Change Curve" (Fig. 3). A single drop in performance during the transition period characterizes the Classic Change Curve – a performance dip. In a typical change program, the performance at the final equilibrium state may differ little from that at the initial equilibrium state, but an effective change program should result in a much better performance than before as the organization settles into its new way of doing things. Organizational performance is a topic in its own right, as it can refer to financial aspects like profitability or nonfinancial aspects like obedience to the law, customer satisfaction, or employee satisfaction and productivity [4]. These expectations can be set by the environmental stakeholders (customers, owners, and community) as well as by the process stakeholders (employees and suppliers) [4]. Despite the metrics used, any improvement effort should aim at a significant performance improvement to cover the cost and risk involved in making the change.

The ease of introducing changes in an organization depends on what defines the organizational capabilities. Initially, the

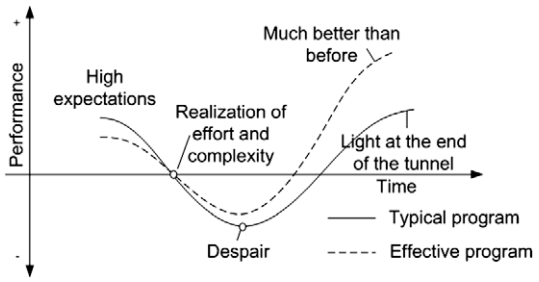

Fig. 3. The Classic Change Curve [66]. The Point of Despair depicts the Death Valley of Change.

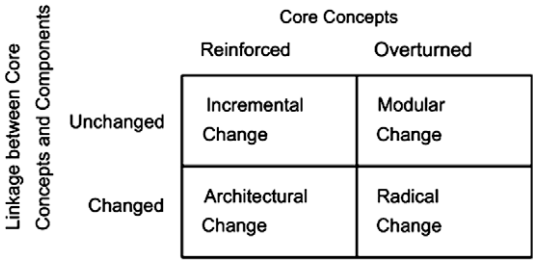

Fig. 4. Technological change framework to classify change types (adapted from [33]).

organizational capabilities are defined by resources (e.g., people, equipment, cash, information, and brands), then by articulated processes and values, and finally by culture, which means that employees start to follow the processes and decide priorities by assumption rather than by conscious choice [17,65]. As long as the capabilities reside in resources, changing organizational capabilities is relatively easy, but when they reside in processes and values – and especially in culture – a change can be very difficult to implement [17].

2.3. Types of technological change

Changes come in many forms and sizes, so a meaningful discussion requires a framework within which the scope and nature of the change can be understood. Henderson and Clark's study on innovations [33] proposes a framework to classify four different types of technological change, or innovation, based on system components and their linkage (Fig. 4). In the case where a change or innovation reinforces the core concepts of the components but leaves the linkage between the components intact, the change is considered incremental. A change that overturns the core concepts at the same time as it modifies the linkage between the components, and the number of components, is considered a radical change. A change that is limited to the components but overturns the core concepts is considered a modular change, while a change that focuses on the linkage between the core concepts and the components is considered an architectural change.

Henderson and Clark [33] demonstrate the relevance of architectural innovations by describing how they can destroy the usefulness of an organization's architectural knowledge. Their study looks at three episodes in the semiconductor photolithographic alignment equipment industry and concludes that, in each case, the organization's dramatic loss of market share resulted at least partly from a failure to respond to architectural innovation effectively. Christensen's studies in component technology [15] and architectural innovations [16] report similar findings in the disk drive industry.

3. Research method

The longitudinal case study reported in this paper provides a single thread analysis of the events in one company, Agricultural Data Processing Centre Ltd. (ADPC) of Finland, from 1998 to 2008. The study started from the observation that the company profits for the given period resembled the typical change curve often presented in the literature. A closer literature survey on the subject uncovered the Elrod and Tippett [26] article on change models, which then formed the scientific basis of the present study. Realizing the limited prior research on change and change resistance in software engineering, the idea of undertaking a descriptive case study [76] of a radical technology change in the software domain emerged, with the overall goal to validate the Classic Change Curve in software engineering empirically. However, rather than just confirming the trajectory of a typical change project unfolding, we also wanted to understand the reasons behind it, which led to the explanatory elements of the study [76]. The study was therefore designed to answer the following research questions:

1. What are the reasons that initiate technology change?
2. How does a software technology change project unfold?
3. What factors affect the change project unfolding?

The study included multiple units of analysis and employed a number of data collection methods. The primary unit of the analysis was the technology changes, focusing on the technical infrastructure, but organizational and management aspects evolved as equally important parts of the analysis, along with people and performance measurements. The data collection of the study started as part of an earlier research project in which the ADPC Corporate Vice President summarized the technology change project to the first author of this paper. This led to a review of the ADPC annual reports for 1997 through 2008, and resulted in the formation of the initial ADPC change curve and timeline. These two artifacts were further developed, with the participation of the Corporate Vice President, and published previously as phase one of the study [52].

The second phase of the study set out to find answers to the above mentioned research questions. All the research questions were addressed in the interviews with company personnel, to identify possible omissions and biases in the collected data, and to confirm a correct understanding and interpretation of the data. Five interviews were conducted with the company Managing Director, Corporate Vice President, the mainframe system manager, and two software specialists, one in the mainframe environment and one in the network server environment. The interview approach has been used previously and with success in critical success factor studies [e.g., 35,50,60] and, in the present study, the approach was selected to get different organizational viewpoints on the topics of interest. The focused interviews [76] were conducted during 1 day, lasted between 1 and 2 h, and followed the same protocol. This comprised of an explanation of the interview goals, noting the option to turn off the audio recorder, and an explanation of the role of the interviewee at ADPC before, during, and after the change project. The interviews focused on three issues, reflecting the research questions: the project start-up decision, the project unfolding, and the factors that affected both the initial decision and the unfolding process. All the interviewees were asked general questions about what initiated the change project, how the project was managed, and what kind of problems they observed during the project. However, many questions were also specific to different stakeholders as, for example, the software specialists provided insight into the daily software development practice while the Managing Director elaborated on the customer and owner relationships. Finally, the interview protocol included context free [29] and open-ended questions so that the interviewees were able to raise any issues that they considered important. The five interview responses converged on the main questions and, since no new major issues were raised, no further interviews were conducted. All the interviews were audio recorded, transcribed, and analyzed with the ATLAS.ti [67] application.

The data gathered from the interviews were complemented with documents from the interviewees' personal archives and company databases, as well as with numerous discussions with the Corporate Vice President. Materials from the personal archives included documents on previous organizational structures within the company, process and guideline development activities, database conversion records, and a proposed list of changes required to get the change project under control. The company databases, on the other hand, proved invaluable in tracking the company performance over time. All these data were analyzed, and ensuring their observation from multiple data sources validated the findings. A literature study was also conducted to confirm the general relevance of the study findings. Finally, the manuscript was reviewed and validated by the Corporate Vice President, who is also a coauthor of this paper.

The threats to validity [76] for this case study have been mitigated via the following tactics. First, construct validity has been addressed by using multiple sources of evidence for our analyses, by establishing chains of evidence, and by having a key stakeholder from the company review the present paper. Second, internal validity has been addressed by looking for similar patterns in different data sets and building rival explanations. The interviewees were used to validate the explanations developed during the review of the annual reports, and the interview transcripts validated many events identified in other interviews. Third, external validity or the generalizability of the study results has been explored by comparing the findings with existing literature and by noting that the goal of the present study was to validate previously established frameworks in a new domain, most notably the model of the Classic Change Curve. Finally, the reliability of the study has been addressed by, for example, documenting the interview questions and the data sources used, and by using pro forma annual reports as the basis of the study.

4. Case study

The software technology change project at the Agricultural Data Processing Centre Ltd. (ADPC) of Finland started at the end of 1998 and was scheduled to finish by the end of 2001. However, it incurred delays and so continued until 2004. The case study covers this entire period and also covers the subsequent sustained process improvement period, a phase triggered by the change project and which continued until the end of 2008, thereby covering a decade of changes at ADPC.

4.1. Background for the technology change

ADPC was formally established in 1986 out of the IT department of the National Insemination Association of Finland. The Association had purchased its first IBM punch card computer in 1958, which was followed by an IBM 1401 in 1965, and then an IBM 360/30 in 1968. In the mid 1980s, ADPC moved from an IBM 4341 to IBM clones and, as they moved into the 1990s, they were using IBM 9021 compatible hardware with the IBM VSE and VM operating systems. Regular maintenance operations were performed until 1998, as the computing power increased and the memory storage techniques improved, but the data collected since the 1950s and the applications developed since the 1960s

remained a central part of the system as the millennium approached. The most substantive prior changes to the system had been the introduction of terminal connections in the early 1980s along with the adoption of the VM operating system, and the move from file-based data storage to a Datacom database in 1991. In 1998, ADPC had 44 employees, 20 of whom were responsible for developing and maintaining the applications that accessed and modified the company's data in the single database.

At this time in 1998, ADPC was owned by seven companies and by associations representing farmers. The customers included all the owners, along with other organizations such as the public administration that maintained the national primary production registers that contained information for all the bovine, pigs, and lambs in Finland. The European Union requires that each member nation maintains a national cattle register to be able to identify, register, and track all the major events in the lives of all bovine animals before they are slaughtered. In Finland, such a register was developed by ADPC and put into national use in 1995 as a first of its kind in the whole of the European Union. The farmers were responsible for reporting all the key events to the registers, data that are then used by breeding and dairy specialists in advising the farmers. Overall, ADPC was operating in a co-operative manner, meaning that it was not expected to maximize profit but rather to service all its customers profitably.

During the 1990s, ADPC faced increasing requests for new functionality for their system, including Internet connectivity and usability improvements. Implementing such changes proved a challenge, as can be typical for a legacy information system [14]. Software development tasks were undertaken by individual developers at ADPC, as a lack of development standards in the company and minimal documentation made it difficult to collaborate. Recruiting new people and outsourcing development also proved problematic due to a lack of necessary competence in the mainframe environment and in the High Level Assembler (HLASM) language used by the company. The company's software development ecosystem had effectively reached an equilibrium, one that it tried to maintain by resisting changes [65].

In 1998, everyone at ADPC agreed that the IBM 9021 clone being used was outdated and needed replacement. Since the mainframe alternative proved not to be financially viable, a decision to change to a Microsoft NT-based network server system was made. The technology change project had four goals: (1) to redevelop the system in a new technical environment; (2) to replace the individual-centric development practices by organization-centric ones; (3) to improve the system usability and functionality; and (4) to reduce the system operating costs.

4.2. The change project

The technology change decision was made at the end of 1998 based on the Managing Director's estimate that the system rewrite would take 3 years and be a 20–25-person year effort. The start of the change project was marked by the introduction of a new project organization, and the Managing Director assumed the role of overall project manager while the Corporate Vice President focused on the existing business and customers. This organization was based on three technology groups to focus on statistical, system, and NT technology development respectively. Each group reported to the Managing Director (Fig. 5). The NT group was responsible for the new system development and was led by a Software Development Manager. Since ADPC had no prior experience of the target platform and of projects of this size, the Board of Directors retained the services of a Management Consultant. The actual system development was carried out as five subprojects that focused on customer sectors and applications. The subprojects were tracked and coordinated by the Managing Director together with the Management Consultant in monthly meetings. The Software Development Manager ran the most important subproject, the cattle register project, while newly hired application domain specialists ran the other four subprojects.

The new system was developed primarily by newly hired developers, though external experts were contracted to develop some key components of the system to speed up the development process, like the TCP/IP based communications between the server and the client applications. A third developer group, the original system developers, was given the responsibility to design the new system and to advise the others in its implementation, and they formed a separate system development group that also assumed the mainframe system maintenance duties. To address the apparent skill shortage in both project management practices and the new development tools, a training program was offered to all project members. The key events of the technology change project and the following sustained process improvement phase, overall a 10-year period, are summarized in Fig. 6.

The Windows 2000 platform was released in early 1999 and ADPC changed the target environment accordingly. In mid-2000, the Software Development Manager resigned. The Managing Director subsequently assumed the cattle register subproject leadership, while a newly hired developer took over the Software Development Manager's prior duties. Standards for version management and coding guidelines were developed in the fall of 2000 and, in the spring of 2001, the quality of the developed components became an issue. This issue was first tackled by establishing a test competence center to acquire tools, practices, and knowledge in

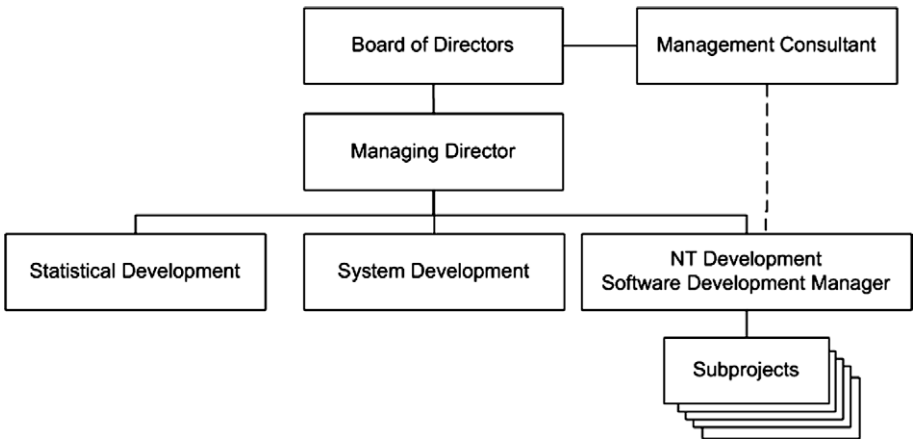

Fig. 5. The ADPC software development organization at the beginning of the change project.

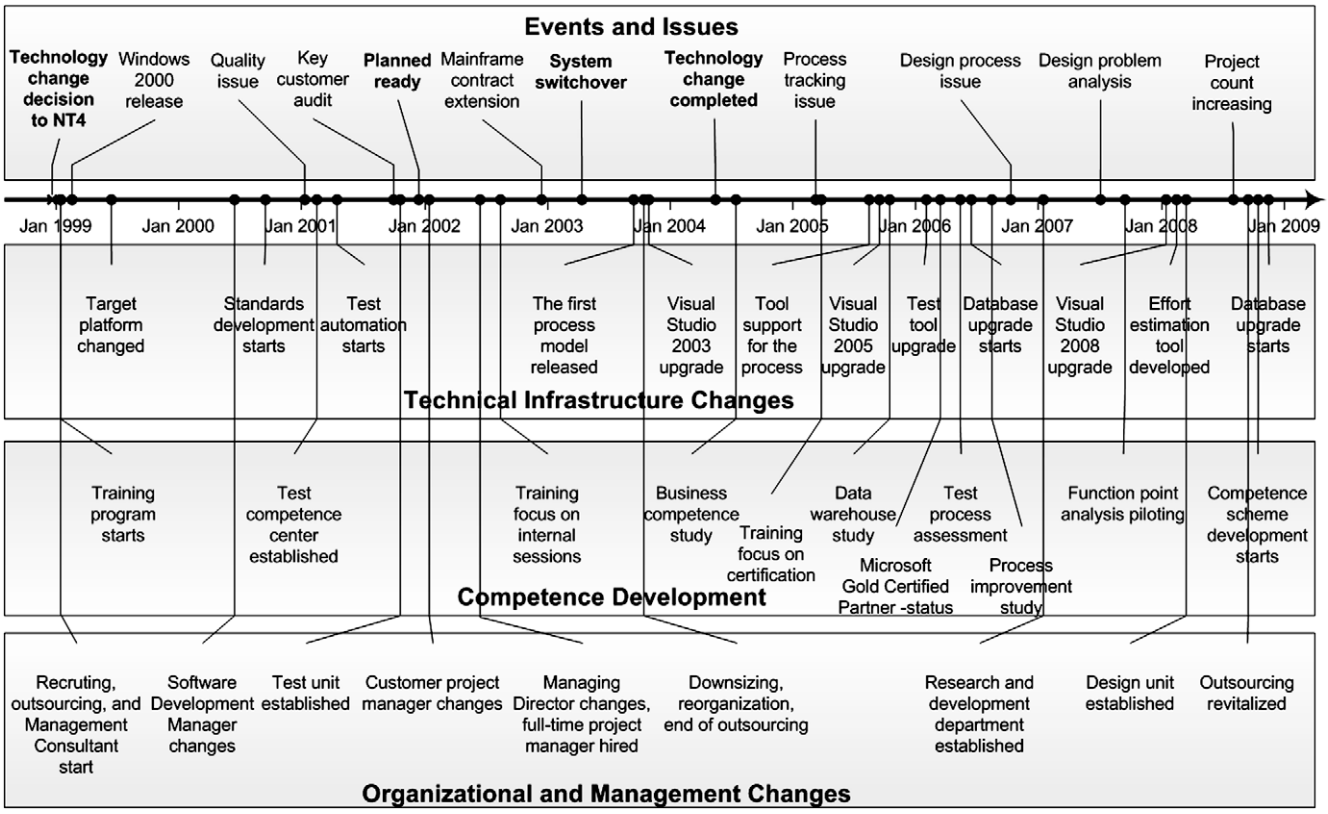

Fig. 6. The major changes at ADPC between the end of 1998 and 2008, chronologically ordered with activity change areas of events and issues, technical infrastructure changes, competence development, and organizational and management changes.

the company. The inevitable increase in the testing effort was balanced by acquiring test automation tools and a test consultant was hired to help establish standard testing practices in the company. The role of testing was formalized in October 2001 by establishing a separate testing unit.

As the original project deadline approached, the end of 2001, the customer for the cattle register project became concerned about the limited progress and arranged for an audit on their subproject. The audit report raised numerous issues and, to get the subproject back on track again, the Corporate Vice President replaced the Managing Director as the subproject manager. The customer accepted a revised project plan with a new completion date in early 2003. Due to similar problems in other subprojects, the Managing Director resigned in June 2002. The Corporate Vice President assumed the Managing Director's role, while the Software Development Manager became the new Corporate Vice President in addition to his previous duties. The Management Consultant was also replaced with a full-time senior project manager who began to coordinate and track the subprojects on a continuous basis. All the leadership changes, along with the establishment of new and realistic project plans, served to motivate the original system developers to work more enthusiastically on the project. At the same time, the new system developers started to get an increasing number of client applications into testing, so the project finally seemed to progress. The last of the major setbacks was encountered at the end of 2002 as the mainframe license contract expired, the 2-month extension incurring ADPC an extra 300,000 Euro cost.

On April 14th, 2003, the new system was put online, 1 week after the old system had been taken offline, and the new system started serving the customers. As the new system became operational, the development process was finalized and fully released in August 2003. Due to severe financial problems, ADPC terminated all outsourcing contracts and introduced a new organizational structure in October 2003 with 18% fewer personnel. The new organizational structure was comprised of four departments – Software Development, System Services, Customer Services and Relations, and Development Projects – as the mainframe development had ceased and the role of statistical software development was reducing. The technology change project was finally considered completed in May 2004, since only at this point in time had all the applications been used and accepted by the users. In addition, the first development tool upgrade to MS-Visual Studio 2003 had been completed to stay abreast with the latest technology change.

4.3. Sustained process improvement

The completion of the technology change project marked a move to sustained process improvement at ADPC, where subsequent changes were implemented mostly in a modular manner. For example, the technical infrastructure (e.g., the development environment, test tools, and database server) was kept up-to-date by upgrading to the latest technologies as they became available, while new tools for tracking the development process and making effort estimates were developed and adopted as needs arose. ADPC likewise sustained process improvement by: participating in research projects on business competences, data warehouses, and process improvement; conducting test process assessment; and piloting function point analysis methods. All the improvement actions also required an investment in training. After the initial training program at the beginning of the change project, the training focus moved to internal training sessions on a monthly basis as the end of the change project approached. After project completion, the training goal was refined to aim at certification and, with the increasing number of developer certifications, ADPC achieved Microsoft Gold Certified Partner status in early 2006. Towards the end of 2008, a formal competence evaluation and development

scheme was started in the company. At the same time, the number of projects was increasing to such an extent that actions were taken to revitalize outsourcing to help resolve resource constraints.

Despite the focus on modular improvements on individual ecosystem elements, architectural changes affecting multiple elements at the same time have not been avoided entirely as, for example, problems with implementing the design process improvements led to organizational changes. The previous Development Projects department was upgraded to a Research and Development department by moving the application domain specialists from Customer Services into it in 2007. Since this did not solve all the problems, a problem analysis was conducted a year later, and a separate design unit was established with a newly appointed manager and moved into the Software Development department. This activated the process improvement effort in the design process area of the company too, as shown in another study [53].

4.4. The technology change project post mortem

ADPC did not do a project post mortem after the technology change project. Now, a decade after the change project started, and as the long term impact of the changes are visible to complement the short term ones, a post mortem is a much more pleasant undertaking. This section examines how the change project unfolded and names the project phases. A closer consideration of the performance dip and improvement in performance is provided, and the overall project outcome is discussed.

4.4.1. Change project unfolding

The ADPC technology change project caused numerous changes that redefined the way ADPC was organized and operated. After the initial project planning problems were resolved, the project focus moved to completing the software development project. However, the change project also started a stream of changes affecting the whole organizational management. These three problem areas – project planning, project management, and organizational management – are summarized before the ADPC change project is discussed from the Classic Change Curve point of view.

The *project planning* problems were due to insufficient planning, leading to resource problems, hasty decisions, and many changes during the project itself. The original system developers considered the plans unrealistic from the very beginning, which resulted in a change resistance that was manifested by low employee efficiency. The project planning problems ended with the introduction of realistic project plans, and the change resistance was reported to have diminished with the change in the Managing Director. The lack of a clear budget, milestones, and plans created a weak basis for the *project management* which, together with the lack of systematic progress tracking, led to excessive costs, delays, and a customer audit. However, the project management problems were not limited to the planning but also covered system and environmental issues. The original system's documentation was not up-to-date, which meant that the new developers had to study an assembly language that they were not familiar with, which slowed down the redevelopment effort. Further, developing a new system while the old one was still in-use led to two parallel systems, two development teams, and increased costs. The environmental factors included, significantly, the year 2000 preparations. The attention to this problem in the software industry and the approaching IT boom led to a shortage of software developers, which impeded the hiring of new employees. This necessitated the use of external consultants and increased the development costs. This also slowed down the development progress. The project management problems were slowly brought under control after the full-time senior project manager was hired, though they persisted to some extent within ADPC until the end of the change project.

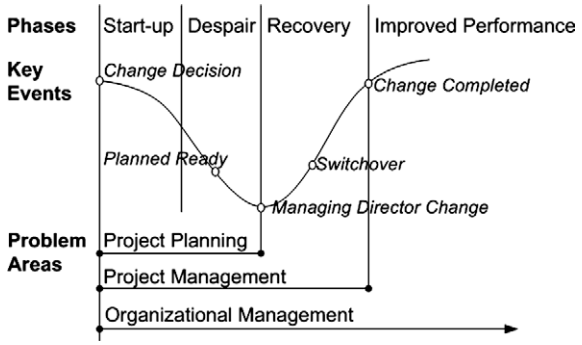

Fig. 7. The Classic Change Curve with the ADPC technology change project phases, key events, and problem areas.

Based on the collected data, *organizational management* had been neglected at ADPC before the start of the change project. However, and as a consequence of the change project, it has been an active area of development within ADPC ever since. From the beginning of the project, it was evident that ADPC had to pay special attention to the employees' learning needs, the technical infrastructure, and the organizational and management structures. Learning new skills is a necessity as new tools, processes, and practices are adopted. At ADPC, the development tools changed roughly every 3 years, and new tools and processes were adopted as needs were identified. The employees were offered a training program at the beginning of the change project and, after exploration with different training schemes, a company-wide development scheme is now under construction. The organizational structure was revised both before and after the change project, and new units and departments were introduced when needs arose. To make these new organizational structures operative, the management structures also changed. Many of the larger changes caused ripple effects, as demonstrated by the quality issues that emerged and were identified in the post mortem: rewriting all the code led to the need for extensive testing, something that was not possible to do with the existing resources, requiring the recruitment of testers. As the number of testers increased, it then became necessary to establish a separate test unit with its own manager.

The duration of the three main problem areas during the project – project planning, project management, and organizational management – are shown in the lower part of Fig. 7 to indicate how they correlate with the Classic Change Curve. The factors affecting the change project outcome – project planning, change resistance, project management, system documentation, parallel systems, environmental issues, learning needs, technical infrastructure, organizational and management structures, and the ripple effect – will be discussed in a broader context in Section 5.3 to show that they are not idiosyncratic to the present study.

The upper part of Fig. 7 shows how the ADPC software technology change project progressed through four phases: start-up, despair, recovery, and improved performance. Even though no exact transition points can be pinpointed: the transition from the start-up to the despair phase started as problems started to emerge; the transition to the recovery phase took place as the project was gotten under control; and the recovery phase was closed as the original performance levels were reached. In the ADPC case, the change project itself was closed as the recovery phase ended, and the performance improvement only started to show properly after the official project closure.

4.4.1.1. Start-up phase. The technology change project started with a performance dip as new skills and tools were acquired and new frameworks, templates, and practices were developed and adapted to the company context. The project experienced problems

associated with low employee efficiency and management changes, problems that deepened the dip to the point where it entered the phase of despair.

4.4.1.2. Despair phase. Despair on the project started to materialize as the original deadline for the project approached. The slip in the project schedule was an early sign of the problems and was compounded by the lack of visible project progress. Even though members from the Board of Directors were not interviewed for this study, it can be assumed that the board members felt some kind of desperation: first, with the outcomes of the customer audit which impugned the whole change project after 3 years of work; second, when they learned that the company made 1,017 KEuro negative result in 2003 when the starting point was 743 KEuro positive result in 1996 (Section 4.4.2); and third, with the need to renew the mainframe contracts at the end of 2002, resulting in an extra 300 KEuro cost.

The despair phase closes with the turning point on the curve, the Point of Despair, when the performance decline ends and positive events start to increase. In the ADPC case, the change of the Managing Director represents the Point of Despair. Even though this action was not the only one, it appears that it was the one that managed to change the course of the project with the introduction of realistic project plans and diminishing change resistance as a result.

4.4.1.3. Recovery phase. Technically, a clear sign of the transition to the recovery phase was the appearance of ready-to-test client applications. Even after the new system was put into operation, the included applications were only all eventually exercised up to 1 year later, since some of the functionality was only required periodically, drawing out the time in this phase.

4.4.1.4. Improved performance phase. After the completion of the technology change project, changes have continued by focusing on individual issues one at a time and in an incremental or modular way. Standard documentation practices and a deputy system have been adopted to avoid reliance on individual developers, the development tools have been updated as new versions have been released, novel approaches like function point analysis and effort estimation have been explored, and emerging issues have been resolved as they have arisen.

4.4.2. Performance dip

The occurrence of a performance dip in a change project has been empirically verified in a team performance study [26]. However, the performance measurement used in that study required the development of two specific survey instruments, something that is not possible in a retrospective case study. Since standard product and process measurements are still somewhat primitive in software engineering, we resorted to the annual pro forma financial statements to study ADPC's performance. As shown in Fig. 8, the net result resembles the Classic Change Curve (Fig. 3) with a performance dip and recovery to the original level. At ADPC, the technology change project was the only major project conducted during this period, aside from the year 2000 preparations, so based on our interviews the performance dip was attributed predominantly to the change project. The quick financial recovery supports this view as it took place as a result of closing the consultancy and mainframe contracts and of the company downsizing in 2003. These items of expenditure caused ADPC ca. 1 MEuro, 0.5 MEuro, and 0.5 MEuro extra costs respectively in 2003, which explains a large part of the 1.8 MEuro increase in net result in 2004. Due to the co-operative origins of the company, it has not aimed at increased profit but in re-establishing the original profit levels as shown after the project closing. ADPC also reimbursed the 2.8 MEuro loans it took during the project by 2008, and the equity ratio rose from 63% in 1998 to 71% in 2008, after reaching a low of 8% in 2003.

4.4.3. Performance improvement

In the ADPC case, the performance improvement expected from a technology change project is supported by some trend data. The ADPC market share in Farm Management Software increased from 20% in 1997 to 64% in 2008 (Fig. 9a) while, in the same period, the proportion of its customers using electronic data channels to report farm data increased from 47% to 80% (Fig. 9b). The transaction errors caused by errors in the data leveled off at 3.0% for the old system, while the new system reached a 1.3% level after a temporary increase to 4.1% in 2003 (Fig. 9c). All these data point to a disruptive phase around 2001–2003. The introduction of a database–based task-tracking system in 2005 made it possible to track task completion data. These data show how the proportion of tasks completed on schedule increased from 31% in 2005 to 76% in 2008 (Fig. 9d).

The adoption of a new version control system also made it possible to track the system size. At the system switchover the new system had 242 K lines of SQL statements and 980 compiled modules. During the first year in operation, the SQL lines increased by 32% and the number of modules increased by 41%. Between 2004 and the end of 2008, the total increase was 161% for SQL statements and 183% for compiled modules, with an annual increase of between 9% to 21% and 11% to 18% respectively. These figures do not suggest any clear trends; rather, they reflect the prevalent foci at different times, such as the development environment change in 2005 leading to less new code, and the unexpected increase in customer projects in 2006 resulting in an increased amount of new code.

4.4.4. Technology change project outcome

The primary reason for the ADPC technology change was the desire to stay in business by switching to an up-to-date technical environment. ADPC managed to do this and also achieved the stated goals of the change project. First, the technical environment was brought up-to-date and, to avoid similar situations in the future, an alert technology switching strategy was adopted. Second, the individual-centric approach to software development was replaced by an organization-centric approach, an approach where knowledge is shared through standardized practices and named deputies. The creation of a competence improvement scheme demonstrates the focus on personnel improvement that is now present within ADPC. Third, the customers have been offered better services both in the terms of product features and development processes, which have become possible through the use of the latest technological innovations. Fourth, a more cost efficient operating environment has been established. The changes started to pay off with the shutdown of the mainframe system and the stabilization of the new system half a year after the switchover. Overall, the benefits of the changes have been realized slowly, as positive trends on each of the technical, organizational, and financial fronts show.

The changes have also raised critical voices. For example, the cost efficiency of the current practices has been debated since the profit level is about the same as 10 years ago, even though the net sales have increased by 30%, and the personnel have increased by 60% in the same period. Also, maintaining the network server system with tens of servers like email, database, web, and name servers along with firewalls and backup servers has proven a challenge for maintenance. In particular, version changes tend to cause ripple effects throughout the other servers, and so the count of maintenance personnel and the required associated competences are now increasing.

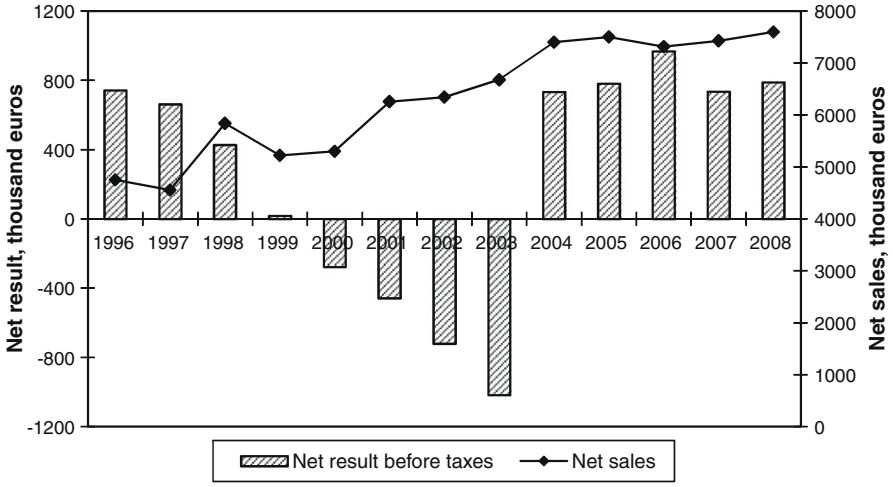

Fig. 8. ADPC net result before taxes and net sales, during the period 1996–2008. The technology change project started at the end of 1998, the system switchover took place in April 2003, and the company downsizing took place in October 2003. The net result peaked temporarily in 2006 due to two unexpectedly large customer projects. The net sales increased during the period mainly due to increased prices.

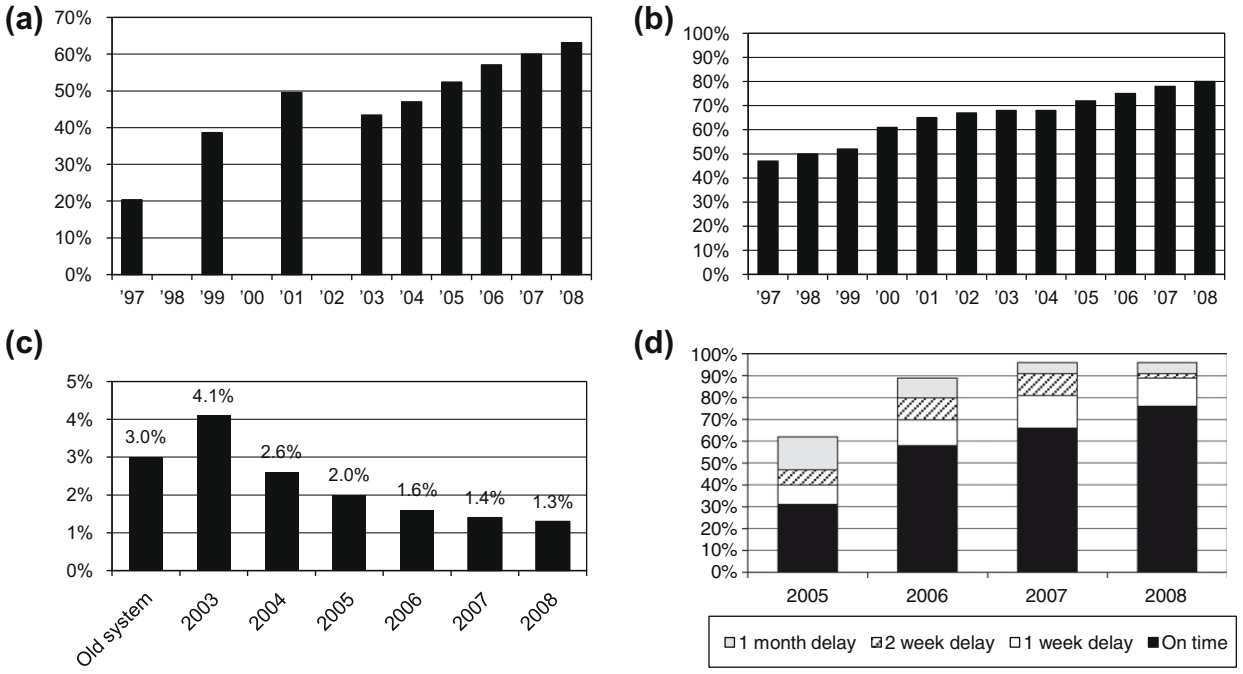

Fig. 9. ADPC performance indicator trends suggest improved performance: (a) increased market share while the market receded from 36,000 to 17,000 farmers (only biannual data was available prior to 2003); (b) increased customer use of electronic services; (c) reduced transaction errors; and (d) improved effort estimation after the introduction of tool support in 2005 (no data was available prior to 2005).

5. Discussion

The previous section presented our case study findings. In Sections 5.1–5.3, the three research questions are re-examined and discussed, based upon these findings, and further elaborated with supporting literature:

1. What are the reasons that initiate technology change?
2. How does a software technology change project unfold?
3. What factors affect the change project unfolding?

The discussion also includes the lessons learned from the study (Section 5.4) and the limitations of the study (Section 5.5).

5.1. What are the reasons that initiate technology change?

At ADPC, the software technology change was initiated due to reaching the limits of the technologies in current use (Fig. 1). The hardware had become old and needed replacement, the mainframe technology was expensive to maintain, and the individual-centric development practices were found risky for the company. A technology change leads to a discontinuity (Fig. 2) as the new technology is adopted and adapted to the company context, and as the skills required to use it are learned. We approached this discontinuity from the Classic Change Curve point of view (Fig. 3), which the ADPC change project followed in principle, barely avoiding the Death Valley of Change. After the system switchover, ADPC

adapted to the new situation through organizational restructuring and downsizing, which started the financial recovery. The closure of the change project also marked the company's return to the technology S-curve after the discontinuity, and ADPC has continued to see steady improvement thereafter on each of the technical, financial, and organizational fronts. Technological changes have not been avoided since the change project completed, but their impact has been kept manageable by increasing their frequency, and changes have been implemented as incremental or modular ones (Fig. 4) whenever possible.

Before the change project, ADPC implemented larger changes to the system roughly once every 10 years (Section 4.1) but, with the introduction of the new system, the change period has been reduced to about once every 2–3 years (Fig. 6). After the system switchover, the development environment was upgraded in 2003, 2005, and 2008, and the database server was upgraded in 2006 and 2008. Testing tools have been developed alongside the development tools, and design function improvements have focused on organizational and process issues so far. Based on the interviews at ADPC, there is a consensus that the company needs to follow an alert technology switching strategy, and technologies are now changed in small projects whenever they are deemed worthwhile and where the production works allow. New technologies are also researched and tracked in an active manner, and experiments are conducted to learn more about them before any adoption decisions are made, as demonstrated with the staged adoption of processes, the effort estimation tool, and the function point analysis.

The frequency of technology change has been reported to vary between companies, just like the performance gained after the change. For example, Christensen [15] reports that some late technology changing companies achieved over seven times the improvement in performance with the recording density of magnetic disks than those who changed early. Christensen explains the performance improvement by modular innovations, or changes, that companies implemented in their systems. Christensen also notes that in five of the seven architecturally defined product generations between 1960 and 1990 in this sector, entrant companies led the changes, and explains this by architectural changes in the product combined with market innovations [16]. Christensen's later studies have led to the notion of disruptive innovations that are often cheaper than mainstream products but that can create new markets and attract mainstream customers as they mature. The Internet is a disruptive innovation that has redefined many business processes, including the ones used by ADPC, and a delayed move to Internet-based applications could well have given the competition an attacker's advantage [27].

Even though much changed during the ADPC change project, two things remained the same: the need for competent software developers and continuity in the customer support personnel. This latter aspect was considered especially important at the time of the system switchover since, despite all the changes, the customers could still talk with the same people they were used to dealing with.

5.2. How does a software technology change project unfold?

The ADPC software technology change project started with an initial performance dip that was followed by a performance recovery reaching roughly the original performance level (Section 4.4). In the absence of comparable performance data for the whole study period, a significantly improved performance cannot be shown, even though the gathered data supports this perception. Overall, the observed performance changes provide an empirical validation of the key characteristic of the Classic Change Curve, with the initial performance decline and the ultimate performance improvement, in a software technology change project. The continued analysis of the case study revealed the four phases of the Classic Change Curve – the start-up, despair, recovery, and improved performance phases – which provide a simple answer to the second research question about software technology change project unfolding. However, to better understand the reasons for the performance dip and the problems of showing performance improvement, a closer look at the literature is needed.

The performance dip has both essential and accidental reasons. The essential reasons derive from the effort required by the typical innovation decision process with the following stages [61]:

1. *Knowledge:* the adopter (an individual or other decision-making unit) is exposed to an innovation's existence and gains some understanding of how it functions.
2. *Persuasion:* the adopter forms a favorable or unfavorable attitude toward the innovation.
3. *Decision:* the adopter engages in activities that lead to a choice to adopt or reject the innovation.
4. *Implementation:* the adopter puts an innovation into use.
5. *Confirmation:* the adopter seeks for reinforcement of an innovation decision already made, or reverses a previous decision to adopt or reject the innovation if exposed to conflicting messages about the innovation.

The actual implementation of an innovation, or the making of a change, involves training and adapting the innovation in the company context, which may require more resources than acquiring the innovation in the first place [51]. In the present radical technology change case study, an extensive training program was needed and all the technical infrastructure, processes, guidelines, frameworks, and templates had to be developed. The accidental reasons, on the other hand, are basically avoidable. In the case study, idiosyncratic problems like project preparation and management, change resistance, and starting the project at same time as the year 2000 changes and IT boom further deepened the performance dip. The role of change resistance in the case study appears important but, due to the retrospective nature of the study, its accurate extent has been difficult to estimate. It is evident though that the change had both winners and losers, as the technical environment changed along with most of the developers, and motivating the original system developers in the project initially failed. We discuss the change resistance further in Section 5.3.

A change project should aim at achieving a significant performance improvement. An easy way to show performance improvement is to rely on qualitative data, like expert interviews. Even though this option was used in the present study, more objective quantitative data was also sought. In the absence of systematic data collection, however, it is difficult to compare the situation before and after a change since so many things can change and impact the change. In the present study, no baseline data had been collected before the change project started but, due to the size of the project and its strategic role within the company, evidence of the project was visible in the company's financial data, and in further data on the employee count, organizational and management structure, technical infrastructure, market penetration, and customer base. It was therefore possible to develop trend data to describe the company progress over time in these various dimensions. The move to the new technical infrastructure then provided increasing insight into the company operations, as data on task completion and system size became available, for example.

Performance measurement is not a new problem, and some software process improvement studies have reported return on investment (ROI) figures. For example, van Solingen [73] summarizes ROI figures for fifteen studies ranging from 1.5 to 19 with an average of 7, while Jones [37] reports to have observed a ROI

between 3 and 30. Changes are much easier to justify with figures like ROI than with the positive trends reported in the present study. The need for clear performance improvement and ROI measures is evident as the changes often involve tangible costs and risks, like the estimated two thirds failure rate for major technology change projects [47] and leadership changing hands in seven out of ten cases [27].

5.3. What factors affect the change project unfolding?

Section 4.4.1 summarized 10 factors that were found important in the ADPC change project: project planning, change resistance, project management, system documentation, parallel systems, environmental issues, learning needs, technical infrastructure, organizational and management structures, and the ripple effect. In Section 5.3.1, these factors are discussed to confirm their relevance outside the specifics of this present study, and a summary of the factors is presented in Section 5.3.2.

5.3.1. Discussion on the observed factors

5.3.1.1. Project planning. The importance of project planning was summarized by the ADPC Corporate Vice President in the interviews as follows: "If we would have understood the whole project better, we could have done many things a lot better – for example, the phasing of the work." This problem sounds familiar to anyone who has participated in a project with insufficient project planning, and the problem can also be found in the literature, reported as a lack of planning [62,70], or simply as problems with planning and estimation [12,41,62]. In general, problems with project planning are often discussed under project management topics in the literature (see below).

5.3.1.2. Change resistance. Due to the post mortem nature of the ADPC case study we have only anecdotal evidence of change resistance and the role of developer motivation with respect to the project's progress. During the change project, the company appeared to operate like an initial-level company in the maturity of its practices, with project management problems rife and the reliance on individuals evident, as highlighted in the interviews with the key people involved. For example, the interviewees reported that "the resignation of the Managing Director improved the working atmosphere", "the hiring of the new project manager shaped up the project", and "one original developer committed to the change project and started working which also started progressing the project". These comments suggest that the developer motivation affected the project progress considerably, which aligns with previous studies that report on how motivation affects project productivity [57], software quality [11], the project's overall success [28], and timely delivery with adherence to budget [31]. The studies of developer motivators on software process improvement efforts report factors like visible successes, available resources, top–down commitment, and bottom–up initiatives [5], while the top three de-motivators have been reported to be the lack of resources, increased workload, and the lack of management direction/commitment [49], as well as time pressure/constraints, and inertia (resistance to new practices) [6]. Many of these factors were also observed in the case study in ways that thus inhibited progress. The passive change resistance reported as most common at ADPC has been studied closely by Sandberg and Mathiassen [63]. Their report described the typical actions used by managers and senior engineers within Ericsson to slow down software process improvement efforts, including: they did not come to meetings – "don't show up"; "answer late"; answer "great, but not now"; and "say nothing." Since change resistance at ADPC arose mainly from the original system developers, the forms of resistance appear a bit different, but behaviors similar to those presented in Table 1 were exhibited. Despite the change resistance during the project, the ADPC employees reported generally high satisfaction levels in their employee surveys undertaken after the project. The long 10.7-year average length of employment at ADPC, with a standard deviation of half a year, further supports the generally positive working atmosphere.

During the mainframe era, the organizational culture at ADPC revolved around a few star programmers who did most of the technical work in the company, supplemented by a couple of other programmers. These star programmers defined what was done and how, thereby characterizing the company culture as "the way we do things around here" [23]. Even though the change project tried to break free from the hero and individual-centric culture, the change project still created its own success stories and heroes. One apparent reason for this is the inherent difficulty of changing a culture. Deal and Kennedy [23] note that: "...we came to recognize that culture is *the* barrier to change. The stronger the culture, the harder it is to change", which fits well with Christensen and Overdorf's [17] claim that capabilities residing in culture are the hardest ones to change. ADPC, however, managed to instill the seed of an organization-centric culture during the change project, and further raised its role through processes, documents, and the adopted practices after the project was completed. This is noted in the interviews: "The biggest cultural change was the move from individual centric working to team based working." Another quote from the interviews also reflects the cultural changes in the company: "Earlier we just started projects – now they are also completed." The problems ADPC faced in starting the software process improvement actions have also been observed in other companies and, for example, Christiansen and Johansen [18] describe the *ImprovAbility* model that projects can use to identify areas that need attention to increase the likelihood of improvement success. The model consists of 20 parameters forming four groups – foundation, initiation, project, and in-use parameters – that include issues like organizational culture, sensing the urgency for the change, available competences, as well as the definition and enactment of roles and responsibilities.

Implementing a change in an organization effectively is not only a technical matter but also involves leadership and vision – "Change is one initiative you just can't delegate" [66]. In the ADPC case, a lack of technical leadership in the target technologies was evident, but a clear vision of the company after the changes was missing. In the interviews this was noted as one of the largest project risks: "The biggest uncertainty [in the project] was the conflict between the illusions of the previous Managing Director and the reality."

5.3.1.3. Project management. Insufficient project planning provided a weak basis for the ADPC project and things started looking better only after realistic project plans were introduced and a full time manager was hired to manage the project. It is not a coincidence that project management is a regular topic in software project risk lists. Typical project management problems include scope changes [39,62], lack of resources [62,70], and inadequate or missing project management methodology [41]. These issues also arose in the case study, aligning with most of the nine project management process areas: integration, scope, time, cost, quality, human resources, communication, risk, and procurement management [58]. The role of basic project management practices is also exemplified in the Capability Maturity Model, where a move from an initial – chaotic – level of process maturity requires the establishment of project and requirements management practices [55]. However, even if all of the basic project management practices had been implemented by ADPC, it is likely that similar problems would still have occurred given that the project was unique in the company: the extraordinary 120 person year size of the project, along with

the change to totally new hardware and software technologies at the same time as the development practices and most of the staff were changing. Sauer et al. [64] report a significantly increased project risk with increasing project size when measured by effort, duration, and team size as well as by changes in project manager and sponsor, schedule, budget, and scope. Looking back at the ADPC project, external help could have assisted the company, as has been reported in other software process improvement efforts in small organizations [e.g., 21,38]. However, this kind of runaway project, exceeding the original schedules and/or budgets by at least 30% [41], are not rare in the software industry [30,41,70], even though Glass [30] notes an increasing reluctance among companies to discuss them openly.

5.3.1.4. System documentation. While ADPC acquired its first stored mass-memory computer in 1968, the system included data from the 1950s, so by 1998 the system had accumulated application complexity from over 30 years and data complexity from over 40 years. Based on the interviews, the system documentation was not up-to-date when the change project started and database analysis revealed that some data fields had been used for different purposes at different times to save programming effort. The ADPC system is a classic legacy information system as the introduction of the IBM System/360 in 1964 subsequently made it possible to run applications on all the operating systems of this family [34]. Hopkins and Jenkins [34] call the redevelopment of such old legacy systems *Brownfield* projects, as the systems are often contaminated by complexity. A key issue with Brownfield projects is the systems analysis, as the systems tend to have far more constraints than requirements. As an example, Hopkins and Jenkins [34] note a project in which the functional requirements comprised 250 pages, the non-functional requirements comprised 80 pages, and each of the 250 interfaces had 40 pages of constraints, totaling about 10,000 pages in all. Since the ADPC case matches the Hopkins and Jenkins [34] description of legacy systems in general, the outdated system documentation can be assumed to have had a detrimental effect on the system redevelopment.

5.3.1.5. Parallel systems. ADPC decided to develop the new system from scratch and kept the old system up and running until the new system was found stable enough to switch off the old system. This strategy required two parallel development personnel and incurred double the system costs. There were not too many alternatives to doing things this way, since all the data had to be transferred from the old system to the new one, and the system service break had to be minimized. In the literature, numerous systematic legacy information system migration strategies can be found [e.g., 9,14,20,22,69], as can a taxonomy of strategies [10]. Based on the Bisbal et al. [10] taxonomy, the ADPC redevelopment approach was a revolutionary one as the new system was developed from scratch using a new hardware and development environment.

The standard advice on process improvement and large system redevelopment projects is to avoid revolutionary approaches and implement the changes incrementally [e.g., 9], even though it has been noted that the incremental migration of a legacy information system may actually increase the overall project risk [10]. In the ADPC case, an incremental approach was subject to practical problems. ADPC moved to the Datacom relational database in 1991 and, as technology change became topical, it was realized that the database communicated only with the associated CASE tool and the interfaces that ADPC had custom developed for the HLASM assembly language that they used. An incremental transition was available through the DB2 database, but this option was not viable financially. Thus, with the available knowledge and resources, an incremental technology transition from the mainframe to a network server environment was not possible. Another option for incremental change would have been to first change the technology and focus on improving the development practices later. However, the individual-based practices seemed inappropriate for undertaking a major system redevelopment effort company-wide. Reversing the order of the changes was likewise not reasonable, as the technology change was used as the motivation for the need to change the practices. It can be speculated that the development of a feasible incremental improvement plan should have been possible but, based on the available information, the organization appeared to have neither the motivation nor the skills to do this at the time.

5.3.1.6. Environmental factors. ADPC made the system redevelopment decision in 1998 when, based on the conducted interviews, the system was already quickly approaching the end of its lifecycle and postponing the renewal was not a realistic option. However, the late 1990s were a busy time in the software industry as companies were addressing the year 2000 problem, and the IT boom was fast approaching. It has been reported that the investment of American companies in information technology jumped more than fourfold between 1995 and 1999, and outlays were thus rising briskly at the time [54]. The first major outsourcing contract in IT is reported to have been made in 1989 with the effect of legitimatizing outsourcing [24], and it was later reported that one of the key reasons for outsourcing was the increased labor costs of permanent hiring [68]. The IT boom attracted many competent developers to the extent that less attractive companies had problems recruiting and external consultants had to be used instead. Since local consultants tend to be more expensive than one's own employees, offshore and more recently nearshore companies have been used to provide external consultancy due to the lower labor costs [40]. After ADPC had started the project, it had little other alternative but to adapt to the prevailing environmental factors, and to use the local and nearshore consultants to get the development resources that they needed.

5.3.1.7. Learning needs. The lack of the required technical competences slowed down the change project in many ways at ADPC. First, the adopted technologies – development tools, languages, practices, etc. – were new to the company. Second, the original system developers' participation in the project was limited in the start-up phase and the newly hired developers had to interpret the assembly code before they could develop the new system. To reduce this need for reengineering, much of the new system specifications were redeveloped from scratch based on the old database structures and user interfaces. Third, application domain specialists were hired to operate as project managers for the application redevelopment subprojects with specification duties. It seems evident though that the lack of software project management experience made these hires ill-suited for the tasks at hand. The lack of required competences within ADPC was demonstrated by one of the interviewees who noted that when they requested help they were told to contact external consultants. The absence of appropriate competences in a technology change project is not uncommon [e.g., 30,41,59], and external specialists can be used temporarily to address the shortages while implementing change and improvements [38,72,74]. However, training one's own specialists is important in the longer-term, and ADPC did succeed in establishing company expertise in software development over time.

5.3.1.8. Technical infrastructure, organizational and management structures, and the ripple effect. In the ADPC case study, changes were observed both in the technical infrastructure and organizational and management structures. In many cases, the changes started from the technical infrastructure but, to institutionalize

the changes, organizational changes were also required. This is well demonstrated by the test improvement effort in which a new organizational unit and manager position had to be established before the test function improvement was properly operationalized. This ripple effect is caused by architectural changes (Fig. 4) that change the linkage between the core concepts of the system; to keep the system balanced, all the connected elements have to adapt to the new situation, which may cause further ripples.

The importance of a systemic approach to software process improvement has been noted in various studies. For example, Perry et al. [56] believe that getting a complete picture of the development process requires the study of organization, process, and technology, while Mathiassen et al.'s [46] study elaborates the four organizational elements that must be changed to achieve lasting effects based on change management theory [3]: process, organizational structure, people, and management. Finally, Zahran [77] provides a definition for software process infrastructure consisting of an organizational and management infrastructure together with a technical infrastructure, and all these aspects can be observed also in the present study. The fact that ADPC had to tackle organizational issues to solve technical ones complements the Beecham et al. [7] finding that low maturity companies are more concerned with technical project problems like documentation, timescales, tools, and technology, while higher maturity companies are more concerned with fundamental organizational problems.

5.3.2. Summary of the factors

Table 2 summarizes the 10 factors affecting the ADPC performance in the change project. All the factors were discussed in the interviews. The following factors were also identified in the company documents that were collected and analyzed: project planning, project management, parallel systems, environmental factors, learning needs, technical infrastructure, and organizational and management structures. The remaining factors were not readily identified in the documents: change resistance, system documentation, and the ripple effect.

5.4. Lessons learned

Section 2 introduced the three frameworks that were used in the study: the technology S-curve, the Classic Change Curve, and the technological change framework. This section summarizes the key lessons learned from the use of these frameworks.

The technology S-curve depicts the performance improvement of a single technology during its lifetime (Fig. 1). The performance improvement is initially slow as essential learning takes place and, after a period of steady improvement, the improvement rate slows down as a limit of the technology approaches. The technology S-curve presents two practical questions for technology adopters and users: do you have the resources required for the learning phase; and do you know the limits of your technology?

The Classic Change Curve (Fig. 3) focuses on the discontinuity experienced when a technology switch occurs (Fig. 2). The new technology is in the learning phase at the same time as the old technology is still being used, and the organization needs to adapt to the changes required to transition to the new technology. The start-up phase of a change project results in an unavoidable performance dip caused by the essential learning and adaptations that are required, but numerous factors can further deepen the dip into the phase of despair. However, as the performance decline ends, either with or without the phase of despair, the recovery phase starts and continues until the original performance is achieved. Only after the transition into the improved performance phase can a change project achieve its fundamental goal – a much better performance than before. The Classic Change Curve presents two practical questions for organizations contemplating a change project: can you afford the performance dip; and can you show a performance improvement that justifies all the costs and risks of the change project? It should also be kept in mind that a project that manages to keep to realistic plans and milestones has no reason to enter the phase of despair, and a project can exit from the phase of despair when the project catches up with or changes the plans and expectations.

The technological change framework defines four types of changes, organized as a two by two grid (Fig. 4). Incremental changes reinforce the core concepts without affecting the linkage between them, while modular changes overturn the core concepts but leave the linkage intact. Architectural changes also reinforce the core concepts but modify the linkage, while radical changes overturn the core concepts at the same time as the linkage is modified. Consequently, the incremental and modular changes affect only the elements that are modified, while architectural and radical changes require adaptations in the whole system. Henderson and Clark study [33] show how architectural changes in technology can destroy the usefulness of the architectural knowledge in an organization. Since architectural knowledge is often embedded in the structure and information-processing procedures of an organization, an architectural change that is made without consideration of its impact on the whole system can lead to a severe ripple effect affecting the whole organization. Thus the technological change framework leads to two practical questions for organizations contemplating a change project: do you understand the type of the change you are about to initiate; and do you understand how it affects the project and the organization?

ADPC has not defined any explicit policies or strategies to tackle the issues suggested by these frameworks. However, a study of the changes that occurred both during and after the change project (Fig. 6), alongside data gathered from the conducted interviews and observations made during the company visits, indicate cultural changes (Section 4.1). In particular, new technologies are now introduced regularly within ADPC, but in a controlled way, and personnel development and training are also constant activities. Rather than exploring the limits of the used technologies explicitly, the benefits of new technologies are evaluated and adoption decisions are taken based on the benefits. Care is also being taken so that the adopted tools and versions offer a transition potential to avoid the need for potentially risky migrations and projects akin to a deep change curve later. Overall, the ADPC focus is now clearly on incremental and modular changes, but radical changes are not avoided entirely, as demonstrated by the appointment of the new design manager to the design unit, moving the unit from one department to another, and starting design process improvement activities at the same time. However, this change was preceded by a 4-month problem analysis phase during which three workshops were organized and participation involved representatives from all the affected departments. The final decision was only made after all the stakeholders had commented on the proposed solution.

Finally, the conduct of the study uncovered two new improvement areas for ADPC. First, the company performance tracking did not serve the needs of the study. The customer satisfaction with the company was not tracked and, considering the co-operative nature of the company, this would seem to make sense to do. Second, ADPC does not yet conduct project post mortems in a systematic way, even though these are accepted as "a simple and practical method for organizational learning" [25].

5.5. Limitations of the study

The main limitation of the present study is the fact that many of the initial events took place over 11 years ago and thus some

Table 2
Factors impacting ADPC's performance during the technology change project. The ADPC Actions and Outcomes columns provide example events from the project.

Factor	Role in a technology change	ADPC actions	Outcomes
Project planning	Project feasibility is studied to explore alternative solutions, costs, and schedules. Project plans are developed together with all the project stakeholders to establish a shared understanding of the project	Top management decided to start the project without an initial feasibility study and a properly documented or realistic project plan. The original developer opinions were ignored in the published plans	Unrealistic expectations led to resource shortage, hasty actions, and many changes that prevented plan driven development. Ignoring the original developer opinions led to change resistance (see below)
Change resistance	Change resistance can vary from passive silence to active project sabotage, as per Table 1, but any form of change resistance can slow down a project	The original developers found the proposed schedules unrealistic and many opted to wait and see during the start-up phase	The original developer change resistance led to a project slow down and a lower quality input for the new system developers and consultants. ADPC had to change the project leaders and develop realistic plans to motive the original developers to work on the project whole-heartedly after the start-up phase
Project management	Basic project management practices help to achieve the project goals on schedule and within budget	No explicit project budget was defined, monthly project meetings were held to monitor progress and schedules, and to coordinate subprojects in the start-up phase	Project delays led to a customer auditing one subproject. ADPC had to change the subproject manager and redo the project planning as well as hire a full-time senior project manager to coordinate the subprojects
System documentation	System documentation helps to understand the system and its structure. If documentation is not up-to-date, all such information has to be interpreted from the system code	The system documentation was not up-to-date, and the original developers had limited motivation and possibilities to participate in the project in the start-up phase. The new development practices and system were documented from the beginning	The new developers had to study the old system assembly code before they could redevelop it. This required much more time than estimated, which led to project delays. ADPC had to engage the original developers in the project to speed it up
Parallel systems	Redeveloping an information system from scratch may require running two systems and personnel in parallel until the new system replaces the old one	A new technical infrastructure was acquired and personnel were hired to develop the new system. The old system was shut down after switchover and the personnel were adapted to the new situation through downsizing	Parallel systems caused increased costs due to double personnel, licensing, and maintenance costs. Terminating the old system contracts, external developer contracts, and downsizing the personnel started the financial recovery
Environmental factors	A company has to adapt to prevailing situation in the environment, such as laws, developer availability, and their costs	The project started in 1998 when the year 2000 preparations in software industry were at their peak and the IT boom was starting. External consultants and nearshoring were used to solve the hiring problem	A failure to hire a talented development team led to the use of external consultants and nearshoring. All these slowed down the development of the technical infrastructure. Use of external consultants also increased the project costs
Learning needs	Employees need to learn how to use new technologies efficiently	The project members were offered a training program from the beginning of the project with a later focus on certification. A company-wide development scheme is currently under construction	ADPC built expertise in selected technologies during the project. The expertise has motivated employees and made ADPC eligible for partnership programs, which offer cost reductions and increased visibility
Technical infrastructure	New technologies (e.g., processes, tools, and practices) must be selected, acquired, and adapted to the organizational context to form the technical infrastructure	The development platform was selected as part of the project preparation and tools were acquired as needs arose. New development practices and system architecture were defined and adopted during the project	Developing software at the same time with its architecture and development practices prevented their efficient use in the project. The same problems had to be solved in different subprojects, leading to inconsistent application architectures and rework
Organizational and management structures	Institutionalizing changes in technical infrastructure may require adapting the organizational and management structures first	Software quality improvement was started by acquiring test tools, then by improving the test practices with a consultant, and finally by establishing a test unit with its own manager	Improving the test practices started from the technology-side but, only after the organizational and management structures became established, did the changes get institutionalized and external consultant use ceased
Ripple effect	Architectural changes in system elements and their linkage require all the affected elements to be adapted to the changes. One change can cascade through many elements through the links – the ripple effect	All the elements of the software development ecosystem were changed in the project leading to many unexpected knock-on and emergent changes	Code rewrites led to extensive testing, the hiring of testers, and finally to the establishment of a test unit. The move from a software development process that was individual-centric to an organization-centric approach led to the hiring of more people and eventually to the establishment of separate design, development, and test units, as well as to the development of a task management system

memories can be inaccurate. This problem is further complicated by the lack of project documentation in the early part of the change project and minimal documentation from the mainframe period. It is also evident that the primary source of the initial information, the annual reports, focuses on positive events even though they are required to include correct information by the law. The data sources lack the customer viewpoint, the former Managing Director, and dissidents. Due to the co-operative nature of ADPC, the net result is not a true measure of the company's success, as customer satisfaction would better reflect the goals of the company.

However, in the absence of systematic customer satisfaction surveys this viewpoint was not available. The Managing Director who started the change project would be a natural person to answer questions about the decision making process and the early part of the project but, due to the limited project success under his command and his exit from the project, this access was not possible. The conducted interviews did not find any dissenting voices, even though not all the employees were happy about giving up the mainframe system. One explanation may be that possible dissidents are no longer working in the company. Finally, the Corporate Vice President participated in this study as a key informant, as an analyst, and as an author. The possible bias resulting from the multiple roles has been addressed by triangulating the central issues from the other data sources, especially in the interviews.

The analysis and indicated causalities should be treated with caution due to the long time span since the actual events and this reporting of them. In particular, the factors impacting ADPC's performance during the software technology change project (Table 2) were validated: (1) by identifying each factor in multiple data sources; (2) by having the Corporate Vice President review them; and (3) by finding general evidence for the factors in the literature. However, the omission of factors is possible since each interviewee approached the interviews from their own perspective, and "CSFs [critical success factors] differ from company to company and from manager to manager" [60].

6. Conclusions

This case study set out to empirically validate the Classic Change Curve on a software technology change project. We approached the study by looking at a 5-year and 4 month legacy information system project in which a 30-year old mainframe system was replaced with a network server system and all the technical infrastructure and employee competences were renewed. In the full 10-year period covered by the study, so also including the years following the closure of the change project, 60% of the personnel and most of the organizational and management structures changed. In addition the organizational culture transformed from an individual-centric one to an organization-centric one, and the technology change strategy became one that actively tracked and adopted new technologies.

Technology change became topical in the company of the case study as the old system reached its technology limits. The system failed to provide the functional and usability improvements that customers requested, the hardware was outdated, and attempts to hire personnel failed due to the aging mainframe system and the assembly language used. The actual change project progressed as a typical change program along the Classic Change Curve with an initial performance dip and the achievement of near original performance levels by its closure. However, looking more broadly over the 10-year period we were able to identify improved performance trends on each of the technical, organizational and financial fronts, even though the lack of systematic collection of project, process, and product metrics made straightforward quantitative before and after comparison impossible. A closer study of the change project unfolding revealed four phases in the Classic Change Curve: start-up, despair, recovery, and improved performance. The actual project course and performance was affected by ten factors: project planning, change resistance, project management, system documentation, parallel systems, environmental issues, learning needs, technical infrastructure, organizational and management structures, and the ripple effect.

The practical implication of the reported case study is that software technology change projects can benefit from an understanding of both the organizational and technology change research. The Classic Change Curve helps companies to set realistic expectations on project progression and, thereby, to increase the likelihood of keeping their change projects under control. The notion of performance also raises the question of suitable organizational performance metrics that should be tracked and set as goals on such change projects. On the individual level, the study confirmed that every change involves a loss for somebody, and change tends to result in change resistance. Managing the loss and mitigating the resistance is a central issue in keeping the personnel productive for the duration of the change project.

On the technological side, an understanding of the technology S-curve and disruptive innovations can help companies to time their technology changes. Tracking the performance of key technologies can help in identifying the approaching limits so that workarounds or technology changes can be prepared for. The dynamics of disruptive innovations, on the other hand, is important to understand so as to recognize them as early as possible and to avoid getting caught out. Finally, the nature of the change should be contemplated from the system concepts and their linkage point of view, as depicted by the technological change framework. Careless changes affecting the system linkage can result in ripples that are difficult to get under control after the fact.

On the theoretical side, this case study underlines the importance of further empirical studies in software development practice. This qualitative study demonstrates the significance of organizational and technological aspects of change within software engineering, but quantitative studies are needed to establish the extent of the observed problems. A retrospective study of this nature has numerous limitations, which makes tracking a live change project from the beginning a very interesting topic for study. Finally, the applicability of the observations in other contexts is unclear. In particular, the use of the net result as a performance measure works only for those projects that affect the financial standing of the whole business unit, and the performance dip enlarges quickly with multiple interdependent projects that start to escalate. Determining what would constitute suitable performance measures to track the progress along the Classic Change Curve and technology S-curve in smaller software process improvement efforts needs further study.

Even though the case study has focused on efforts to keep abreast with the latest technology changes in one company, we believe these findings have broader applicability. For example, IBM's development of a new system in the 1960s, the System/360, was also a significant investment in a technology that was not properly known to the developers at the time but proved to be a great success. Bob Evans, the IBM line manager responsible for this so-called "You bet your company" project, summarized it in 1966 as follows: "[the 360] was a damn good risk, and a lot less risk than it would have been to do anything else, or to do nothing at all" [75]. A timely software technology change, coupled with the ability to navigate the ups and downs of its project path successfully, can open up the way to technical leadership.

Acknowledgements

We would like to thank the personnel and the management of the Agricultural Data Processing Centre Ltd. of Finland for participating in the interviews, and for providing answers to numerous follow-up questions on a variety of topics, such as the history of the computer models used over time at ADPC. We are also grateful to ADPC for letting us share all these details from their software technology change project with the wider software engineering community. Finally, we would like to thank the anonymous reviewers of an earlier version of this paper for their constructive suggestions for improving the paper.

References

[1] W.S. Adolph, Cash cow in the tar pit: reengineering a legacy system, IEEE Software 13 (1996) 41–47.
[2] I. Allison, Y. Merali, Software process improvement as emergent change: a structurational analysis, Information and Software Technology 49 (2007) 668–681.
[3] L.M. Applegate, Managing in an information age: transforming the organization for the 1990s, in: R. Baskerville, S. Smithson, O.K. Ngwenyama, J.I. DeGross (Eds.), Transforming Organizations with Information Technology, Elsevier, 1994.
[4] A.A. Atkinson, J.H. Waterhouse, R.B. Wells, A stakeholder approach to strategic performance measurement, Sloan Management Review 38 (1997) 25–37.
[5] N. Baddoo, T. Hall, Motivators of software process improvement: an analysis of practitioners' views, Journal of Systems and Software 62 (2002) 85–96.
[6] N. Baddoo, T. Hall, De-motivators for software process improvement: an analysis of practitioners' views, Journal of Systems and Software 66 (2003) 23–33.
[7] S. Beecham, T. Hall, A. Rainer, Software process improvement problems in twelve software companies: an empirical analysis, Empirical Software Engineering 8 (2003) 7–42.
[8] K. Bennett, Legacy systems: coping with success, IEEE Software 12 (1995) 19–23.
[9] A. Bianchi, D. Caivano, V. Marengo, G. Visaggio, Iterative reengineering of legacy systems, IEEE Transactions on Software Engineering 29 (2003) 225–241.
[10] J. Bisbal, D. Lawless, W. Bing, J. Grimson, Legacy information systems: issues and directions, IEEE Software 16 (1999) 103–111.
[11] B.W. Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[12] B.W. Boehm, Software risk management: principles and practices, IEEE Software 8 (1991) 32–41.
[13] J.L. Bower, C.M. Christensen, Disruptive technologies: catching the wave, Harvard Business Review 73 (1995) 43–53.
[14] M. Brodie, M. Stonebraker, Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach, Morgan Kaufman, San Francisco, 1995.
[15] C.M. Christensen, Exploring the limits of the technology S-curve. Part I: component technologies, Production and Operations Management 1 (1992) 334–357.
[16] C.M. Christensen, Exploring the limits of the technology S-curve. Part II: architectural technologies, Production and Operations Management 1 (1992) 358–366.
[17] C.M. Christensen, M. Overdorf, Meeting the challenge of disruptive change, Harvard Business Review 78 (2000) 66–76.
[18] M. Christiansen, J. Johansen, ImprovAbility(TM) guidelines for low-maturity organizations, Software Process: Improvement and Practice 13 (2008) 319–325.
[19] L. Coch, J.R.P.J. French, Overcoming resistance to change, Human Relations 1 (1948).
[20] M. Colosimo, A.D. Lucia, G. Scanniello, G. Tortora, Evaluating legacy system migration technologies through empirical studies, Information and Software Technology 51 (2009) 433–447.
[21] K.C. Dangle, P. Larsen, M. Shaw, M.V. Zelkowitz, Software process improvement in small organizations: a case study, IEEE Software 22 (2005) 68–75.
[22] A. De Lucia, R. Francese, G. Scanniello, G. Tortora, Developing legacy system migration methods and tools for technology transfer, Software: Practice and Experience 38 (2008) 1333–1364.
[23] T.E. Deal, A.A. Kennedy, Corporate Cultures: The Rites and Rituals of Corporate Life, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.
[24] J. Dibbern, T. Goles, R. Hirschheim, B. Jayatilaka, Information systems outsourcing: a survey and analysis of the literature, SIGMIS Database 35 (2004) 6–102.
[25] T. Dingsøyr, Postmortem reviews: purpose and approaches in software engineering, Information and Software Technology 47 (2005) 293–303.
[26] P.D.I. Elrod, D.D. Tippett, The "Death Valley" of change, Journal of Organizational Change Management 15 (2002) 273–291.
[27] R.N. Foster, Innovation: The Attacker's Advantage, Summit Books, New York, 1986.
[28] S.A. Frangos, Motivated humans for reliable software products, Microprocessors and Microsystems 21 (1998) 605–610.
[29] D.C. Gause, G.M. Weinberg, Exploring Requirements: Quality BEFORE Design, Dorset House Publishing, New York, 1989.
[30] R.L. Glass, Software Runaways: Lessons Learned from Massive Software Project Failures, Prentice Hall, New Jersey, 1998.
[31] T. Hall, H. Sharp, S. Beecham, N. Baddoo, H. Robinson, What do we know about developer motivation?, IEEE Software 25 (2008) 92–94
[32] T.R. Harvey, Checklist for Change: A Pragmatic Approach to Creating and Controlling Change, Allyn and Bacon, Boston, MA, 1990.
[33] R.M. Henderson, K.B. Clark, Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms, Administrative Science Quarterly 35 (1990) 9–30.
[34] R. Hopkins, K. Jenkins, Eating the IT Elephant: Moving from Greenfield Development to Brownfield, IBM Press, New Jersey, 2008.
[35] M.-L. Huotari, T.D. Wilson, Determining organizational information needs: the critical success factors approach, Information Research 6 (2001).
[36] Intel Innovation, Moore's Law. <http://www.intel.com/technology/mooreslaw/>, 09.08.09.
[37] C. Jones, The economics of software process improvement, Computer 29 (1996) 95–97.
[38] K. Kautz, Software process improvement in very small enterprises: does it pay off?, Software Process: Improvement and Practice 4 (1998) 209–226
[39] M. Keil, P.E. Cule, K. Lyytinen, R.C. Schmidt, A framework for identifying Software project risks, Communications of the ACM 41 (1998) 76–83.
[40] D. Kilroe, Outsourcing IT: transferring your assets abroad [software engineering], Engineering Management Journal 6 (2005) 31–33.
[41] KPMG Ltd, Runaway projects: causes and effects, Software World (UK) 26 (1995) 3–5.
[42] E. Kubler-Ross, On Death and Dying, Touchtone, New York, 1969.
[43] C.Y. Laporte, S. Trudel, Addressing the people issues of process improvement activities at Oerlikon aerospace, Software Process: Improvement and Practice 4 (1998) 187–198.
[44] K. Lewin (Ed.), Group Decision and Social Change, Henry Hold, New York, 1952.
[45] J. Mariotti, Troubled by resistance to change?, Industry Week 245 (1996) 30
[46] L. Mathiassen, O.K. Ngwenyama, I. Aaen, Managing change in software process improvement, IEEE Software 22 (2005) 84–91.
[47] R. Maurer, Beyond the Walls of Resistance, Bard Press, 1996.
[48] W.W. Menninger, The meaning of morale: a peace corps model, in: D.P. Moyniham (Ed.), Business and Society in Change, American Telephone and Telegrapch Co., New York, 1975.
[49] M. Niazi, M.A. Babar, N.M. Katugampola, Demotivators of software process improvement: an empirical investigation, Software Process: Improvement and Practice 13 (2008) 249–264.
[50] M. Niazi, D. Wilson, D. Zowghi, Critical success factors for software process improvement implementation: an empirical study, Software Process: Improvement and Practice 11 (2006) 193–211.
[51] U. Nikula, Introducing basic systematic requirements engineering practices in small organizations with an easy to adopt method, Doctoral Thesis, Department of Information Technology, Lappeenranta University of Technology, Lappeenranta, Finland, 2004.
[52] U. Nikula, C. Jurvanen, O. Gotel, D.C. Gause, From technology migration to organizational culture change, in: 11th International Workshop on Learning Software Organizations, Fraunhofer IESE, Oulu, Finland, 2009, pp. 7–19.
[53] U. Nikula, P. Oinonen, L. Hannola, Extending process improvement into a new organizational unit, in: 20th Australian Software Engineering Conference, IEEE Computer Society, Gold Coast, Australia, 2009, pp. 267–276.
[54] S.D. Oliner, D.E. Sichel, The resurgence of growth in the late 1990s: is information technology the story, The Journal of Economic Perspectives 14 (2000) 3–22.
[55] M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis (Eds.), The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading, MA, 1995.
[56] D.E. Perry, N.A. Staudenmayer, L.G. Votta, People, organizations, and process improvement, IEEE Software 11 (1994) 36–45.
[57] J.D. Procaccino, J.M. Verner, K.M. Shelfer, D. Gefen, What do software practitioners really think about project success: an exploratory study, Journal of Systems and Software 78 (2005) 194–203.
[58] Project Management Institute, A Guide to the Project Management Body of Knowledge, fourth ed., Project Management Institute, Newtown Square, PA, USA, 2008.
[59] A. Rainer, T. Hall, A quantitative and qualitative analysis of factors affecting software processes, Journal of Systems and Software 66 (2003) 7–21.
[60] J.F. Rockart, Chief executives define their own data needs, Harvard Business Review 57 (1979) 81–93.
[61] E.M. Rogers, Diffusion of Innovations, fourth ed., The Free Press, New York, 1995.
[62] J. Ropponen, K. Lyytinen, Components of software development risk: how to address them? A project manager survey, IEEE Transactions on Software Engineering 26 (2000) 98–112.
[63] A.B. Sandberg, L. Mathiassen, Managing slowdown in improvement projects, IEEE Software 25 (2008) 84–89.
[64] C. Sauer, A. Gemino, B.H. Reich, The impact of size and volatility on IT project performance, Communications of the ACM 50 (2007) 79–84.
[65] E.H. Schein, Organizational Culture and Leadership, second ed., Jossey-Bass Publishers, San Francisco, 1992.
[66] D.M. Schneider, C. Goldwasser, Be a model leader of change, Management Review 87 (1998) 41–45.
[67] Scientific Software Development, ATLAS.ti – The Knowledge Workbench. <http://www.atlasti.de/>, 25.09.09.
[68] S. Slaughter, S. Ang, Employment outsourcing in information systems, Communications of the ACM 39 (1996) 47–54.
[69] H.M. Sneed, Planning the reengineering of legacy systems, IEEE Software 12 (1995) 24–34.
[70] Standish Group, The CHAOS Report. <www.standishgroup.com>, 11.06.09.
[71] D. Stelzer, W. Mellis, Success factors of organizational change in software process improvement, Software Process: Improvement and Practice 4 (1998) 227–250.
[72] D. Stelzer, W. Mellis, G. Herzwurm, Technology diffusion in software development processes: the contribution of organizational learning to software process improvement, in: T.J. Larsen, E. McQuire (Eds.), Information

Systems Innovation and Diffusion: Issues and Directions, Idea Group Publishing, London, UK, 2003, pp. 297–344.
[73] R. van Solingen, Measuring the ROI of software process improvement, IEEE Software 21 (2004) 32–38.
[74] H. Wickberg, A. Dorling, Learning to improve – the essential incredients, in: ISCN'97 Conference on Practical Improvement of Software Processes and Products, Budapest, Hungary, 1997, pp. 1–12.
[75] T.A. Wise, I.B.M'.s $ 5, 000, 000, 000 Gamble, Fortune (1966) 118–124.
[76] R.K. Yin, Case Study Research: Design and Methods, third ed., SAGE Publications, Thousand Oaks, CA, 2003.
[77] S. Zahran, Software Process Improvement: Practical Guidelines for Business Success, Addison-Wesley, Harlow, England, 1998.