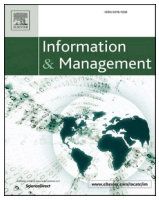
Information & Management 58 (2021) 103407

Contents lists available at ScienceDirect

Information & Management

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



Heroes, contracts, cooperation, and processes: Changes in collaboration in a large enterprise systems project

Kari Smolander [a], Matti Rossi [b,*], Samuli Pekkola [c]

[a] Lappeenranta University of Technology, Department of Information Technology, P.O.Box 20, 53851, Lappeenranta, Finland
[b] Aalto University School of Business, Department of Information and Service Management, Finland
[c] Tampere University, Department of Business Information Management and Logistics, Finland

ARTICLE INFO

Keywords:
Enterprise systems
Software development
Large-scale systems

ABSTRACT

Enterprise systems are developed and tailored in large, long-term projects, sometimes spanning for decades, whereby a network of parties comprising customer and developer organizations, subcontractors, and consultants work together to deliver a successful system. This collaboration is complex; the network and the operating environment are in a constant flux, which creates conflict and challenging situations. Collaboration and ways of working evolve through various crises, internal and external incidents, and project phases. This means that project management practices, communication patterns, contracts, and ultimately personal relationships change.

This longitudinal, qualitative, single case study analyzes a 20-year-old enterprise systems development project, whereby different incidents and crises initiated changes to collaboration practices and the drivers for collaboration. We identified four collaboration modes — contract mode, cooperation mode, personified mode, and process mode — each of which was the main driver in different development circumstances. As a key contribution, we propose the seed of a mid-range theory that provides heuristics for responding to different types of crises that might occur while developing large-scale systems.

1. Introduction

Contemporary enterprise systems development involves partnership between the customer organization, the developer organization, and their departments and subunits such as the IT department and business units that require a new system. Quite often, the systems are developed—or otherwise, a premade package is configured, tailored, and integrated—by a group of development organizations, each contributing adequate domain knowledge, technical expertise, skills, and infrastructure [1,2]. In this network of organizations, there can be any number of subcontractors and parallel projects. All these relationships put extra demand on the team's ways of working and collaboration practices.

Collaboration practices and relationships are not static or rigid. The dynamics and practices of enterprise systems development evolve over the course of various crises, incidents, and project phases. This means that, for example, project management practices, communication patterns, contracts, and ultimately personal relationships might change, whether intentionally or unintentionally. These changes disrupt the system's development, as they cause uncertainties and discontinuities. Enterprise systems development is understood here to exhibit the same issues as other types of information systems (IS) development. The development process is complex, nonroutine, and uncertain [3] and requires close coordination between heterogeneous sets of stakeholders ([4], [5]), with the added complications of long implementation phases and a critical impact on the user organization's health [6]. However, quite little is known about how control dynamics change during a project, how control evolves over time, and what the consequences of those changes are [3]. Some examples of research in this area include the work of Newell et al. [7] and Nandhakumar et al. [8], but their focus is on short-term changes and survival strategies rather than on long-term, evolving relationships. Even in studies on construction-related mega projects such as next-generation nuclear power plants [9] it is assumed that there is a deliberately chosen governance model used to guide the project. There are studies on contract changes between parties but there the focus is on renewing or altering the contracts, not on their evolution over time [10].

These issues motivated us to study how collaboration and control ([4], [11]) between software vendors and clients have evolved over time. As there are no existing theories on the evolution of collaboration

* Corresponding author.
E-mail addresses: kari.smolander@lut.fi (K. Smolander), matti.rossi@aalto.fi (M. Rossi), samuli.pekkola@tuni.fi (S. Pekkola).

https://doi.org/10.1016/j.im.2020.103407
Received 21 January 2019; Received in revised form 24 November 2020; Accepted 28 November 2020
Available online 7 December 2020

in large-scale and long-term enterprise systems projects, we chose an inductive theory-forming approach for our study. Using the grounded theory, we examined an enterprise systems development process that lasted more than 20 years within a large, industrial enterprise. We interviewed the main participants of the development and analyzed the events that occurred during the development process. We focused on these research questions:

How does collaboration in large-scale enterprise systems development change between main development partners?
What kind of collaboration modes exist?

This focus guided our data analysis and helped us understand project collaboration and control over time. As a result, we propose a middle-range theory [12–14] on how collaboration changes within enterprise systems development.

In this article, we describe a longitudinal case whereby the forms of collaboration changed over time, in response to both external shocks and internal issues related to project organization and performance. From these observations, we identified collaboration patterns in the development of an enterprise system over its lifecycle, from concept to near retirement. Our findings illustrate four different collaboration modes—contract mode, cooperation mode, personified mode, and process mode—each of which is a main driver of development under different circumstances. We explain how different kinds of triggers influence and change the mode of collaboration. This knowledge is valuable to those who wish to improve interorganizational practices for enterprise systems development and seek a better theoretical understanding of enterprise systems development. The results of this study make collaboration planning possible in enterprise systems development and aid the mitigation of unexpected project incidents and their effects.

The paper is organized as follows. First, a brief outline of the research area is presented. Second, the research process, data collection, and analysis methods are explained. Third, the case, its timeline, and related incidents are described. Fourth, the case is analyzed, and the collaboration modes are discussed. Finally, the case and its findings are brought to a larger context to discuss their novelty and relationship to the previous literature.

1.1. Research area

Enterprise systems, the most well-known examples of which are enterprise resource planning (ERP) systems, aim to integrate information flows across the organization to increase the organization's competitiveness [6,15]. In this section, we briefly present studies on enterprise systems development, specifically from the perspective of collaboration and control between participating stakeholders and user organizations. As our research approach is purely inductive, this section will not produce a research framework. Instead, we will briefly position our research within the context of other approaches.

Enterprise systems integrate a company's core business processes [16,17]. They are designed to automate the flow of information, materials, and financial resources [18,19] within a company and across a supply chain or network. In the past, they were mostly developed in-house as custom IS. This practice changed when there was a dire need to renew systems in the anticipation of Y2K. Packages at various levels of maturity and for different domains were provided by vendors, such as Baan and SAP [6], as either general-purpose enterprise systems or industry-specific packages [20]. The implementation of systems requires a number of stakeholders from various organizations to collaborate in a complex development network ([4], [11,21]). This form of collaboration is prone to errors and sometimes fatal misunderstandings [22].

The implementation of enterprise systems is studied predominantly through the lens of critical success factors (see, for example, Shaul and Tauber [23], [24]), and Holland and Light)[25]. In these studies, enterprise systems development is often perceived as a linear process with a specific start and end point and a set of separate phases, whereby certain conditions must be met. Robey et al. [26] suggested already in 2002 that instead of linearity, a continuous dialectical learning process is at play and should be studied over time.

Despite the large body of success factor studies, the implementation and adaptation of enterprise systems is prone to failure [8,27–29]. One explanation may be the complex and collaborative nature of ERP development [3,23]. Although these collaborative practices are easily disturbed [22], the different mechanisms, patterns, and changes that cause the disturbance have rarely been studied [30,31]. In this study, we refer to collaboration as a practice whereby at least two parties work together to achieve a common goal or cocreate value in the form of system enhancements [32].

In the IS field, it is assumed that development goals can be achieved through controlled and coordinated development activities ([4], [33]; [22]). For instance, there is a vast amount of literature on controlling outsourced and offshoring system development. Wiener et al. [3] provide a comprehensive review of the studies on control in information systems projects. Kirsch et al. [34] present an example of the notion that control in collaboration is something that is first chosen and then exercised. Similarly, Gulati et al. [35] and Juell-Skielse et al. [36] discuss different types of fixed agreements among organizations that frame the form of collaboration. Ward et al. [31] concretize this by studying the internal stakeholder's power-interest-rights in implementing ERP. Tiwana and Keil [37] study is notable, as they found that attempted control and realized control are disconnected, particularly with regard to controlling external work during enterprise systems implementation.

Survey-based studies on control modes are particularly limited in their relevance to long-term projects, whereby the mode of collaboration must be renegotiated and changed during the course of the project. The control dynamics literature, coined by Wiener et al. [3], provides examples of cases where critical incidents that occurred over time changed the control mode. Gregory et al. [38] develop the idea of control balances that change over time and emphasize the shared understanding between different parties and stakeholders (see also [31]). Gopal and Gosain (2012) highlight that while there is a great deal of research on control mode choice, there is a gap in the current understanding of the effect of organizational controls on project performance. Our longitudinal study attempts to fill this gap and seeks to understand how collaboration changes over an extended period of time, particularly in response to external events that force changes to the control arrangements.

In enterprise systems development, numerous specialists and stakeholders from different organizations collaborate, interact, and influence each other within a development network [11,39,40]. The different stakeholders have been listed, but mostly at a very high level [41]. Chang et al. [1] studied the mechanisms for controlling consultants in a distributed ERP implementation setting, while Levina and Vaast [42] Rosenkranz et al. [43], and Yeow et al. [44] focused on the communication between parties during development, from a boundary-spanning perspective.

Collaboration has been studied mostly from the perspectives of project work, project management [45,46], knowledge dissemination [47–49], or in a certain development phase [31]. All these studies provide snapshots of the development projects, not long-term relationships. Some example cases that focus on longitudinal activities include the studies by Levina and Vaast [50] and Lyytinen and Newman [51], but the focus of these works is not on changes in collaboration over time.

Previous research on enterprise systems development has mostly assumed that collaboration practices or collaboration modes, are decided in a prestudy phase and then stay the same through the entire development life cycle until the system is released for use, transferred to maintenance, or abandoned. However, stories and reports from large-scale development projects provide a more complicated view of the

emergence of systems through a disorganized process [38,52]. In most cases, the development effort is restarted several times, and the collaboration mode is renegotiated each time [8]. Because there are no studies or theories on the evolution of collaboration in large-scale information systems projects, there is a need to understanding how collaboration and control evolve over extended periods of time, as these are the conditions in which most of these systems are developed and evolve.

2. Research process

2.1. The case: birdie at factory

Factory is a global manufacturer of materials and common goods. Its annual turnover is more than 8 billion euros. It has operations, manufacturing, and sales on all continents. At the beginning of the 1990s, Factory realized that it needed to renew its sales and logistics systems. It was decided that a fully customized enterprise system for sales and logistics would be built to replace several legacy systems and to overcome the problem caused by year 2000. This system came to be called Birdie.

Birdie is an enterprise-wide system that includes sales, logistics, and production planning components that are fully integrated and used across the globe. The system is highly distributed, running at more than 50 sites and communicating through asynchronous messaging. Birdie is also fully integrated into enterprise-wide administrative systems and manufacturing systems as well as machinery at each manufacturing site. SAP R/3 is the backbone of enterprise-wide administration and control functionalities such as accounting. Each manufacturing site has its own set of manufacturing execution systems that are integrated with Birdie. Birdie includes a full set of logistics functionalities; however, over time, an increasing number of external logistics systems have been integrated to Birdie. Now, after more than 20 years, many of the logistics functionalities have moved outside of Birdie.

2.2. Interpretive and inductive research approach

Qualitative research methods are essential when human behavior, organizations, and management are studied in their real-world context. We have chosen the grounded theory, originally developed by Glaser and Strauss in 1967, as the research method, due to its ability to inductively reveal the essence of real-world action [53]. As an inductive research method, it is suitable for approaching complex organizational phenomena [54]. Enterprise systems development includes both a technical and a social component, which emphasizes understanding the stakeholders and their interactions, and the effect of the technical issues that the stakeholders face.

The objective of a grounded theory study is to construct a theory as a collection of categories that detail the subject of the research. This theory can be substantive (i.e., pertain to a specific area of the study) or formal (i.e., be general and conceptual) [53]. By acknowledging this distinction, we consider our result substantive, or contextual [14], which means that it details the subject and is transferrable, rather than generalizable [55]. Gasson [55]. It further describes that "a substantive theory is generated when the researcher can define a core conceptual category in the data and identify key patterns of relationships between the various theoretical and conceptual categories that apply across data samples." In our study, *collaboration change* emerges as the core category. It is described through project incidents and the properties of different collaboration modes that are formed and altered by project incidents and changing circumstances.

The grounded theory approach is particularly suitable for dealing with phenomena that are not well understood and that require a better theoretical explanation that is grounded in observations. Complex organizational processes, such as collaboration and how it changes and is changed in large-scale enterprise systems development, exemplify such an area. In our study, we deemed grounded theory suitable for several reasons. First, we could not identify a theory that could explain the dramatic changes in collaboration in the observed project. One of the researchers was very familiar with the project and its 20-year long history. We were therefore aware of different changes in collaboration and their nature. The grounded theory as a method uses relevant literature and theories, but this is done *after* an emergent theory has been identified from the data [55]. The purpose of the literature is to relate the findings to similar fields or situations. The grounded theory is most useful when no applicable theories exist or when the theories cannot adequately explain the phenomenon. In the beginning of our study, we were not able to identify any specific theory that could explain well all the essential project incidents and changes in collaboration over time. Therefore, we deemed, based on our pre-understanding of the project and its history, that an a priori theory would have constrained our interpretation and explanation too much. Second, we expected that we would encounter many different perspectives on the project events, their importance, and their effects. Therefore, we concluded that it is essential to use a methodology that incorporates constant comparison and interpretation. Third, all the researchers were proficient in using the grounded theory. Based on our combined experience, research contexts, and interests, we saw grounded theory as a feasible research methodology for the given purpose.

A middle-range theory (Merton, 1949) is another way to describe the use and purpose of grounded theory. Middle-range theories attempt to predict and explain only a subset of all organizational phenomena [56]. They make different assumptions about organizations, consider priorities differently, and lead to practice prescriptions that are different from other middle-range theories. Each middle-range theory is based on unique research strategies and tactics (ibid.). A middle-range theory can partially explain the phenomena in different domains [13]. Those who use such theories seek to tell a causal story rather than the full story, and they acknowledge that it cannot explain everything (ibid.). Our purpose here is similar. We attempt to explain and abstract how and why collaboration changed over a long period of time in a particular context, first during the development project and then later during system maintenance. In this sense, we are building a middle-range theory for the context of large-scale and long-term project collaboration.

Fig. 1 shows the main phases of the research process. They are presented sequentially for the sake of clarity, although, following the nature of grounded theory, the analysis tends to happen in parallel with the coding, as the theoretical understanding gradually improves. The use of both axial and selective coding required frequent discussions between the three researchers to confirm the interpretations and to provide fresh theoretical views to advance the analysis. Their use also required going back and forth from the theoretical conclusions to the data, and vice versa, to confirm the theoretical conclusions and to aid the naming of the codes, which explains the cyclical structure in Fig. 1.

2.3. Data collection

The data were collected through theme-based interviews between February and May 2013. The data collection began with discussions with our contact from senior management within the user organization. The research goals were briefly presented to the contact to identify the right interviewees in the user organization. In general, the snowball technique [57] was used. To select the interviewees, we sought out those who had important responsibilities and experiences during various parts of the enterprise systems project and maintenance period.

The interview questions were open-ended, which focused on the interviewee's experiences during the enterprise systems project. The interviews included only a few general questions, which then led to more detailed discussions that depended on the interviewee and their background and answers.

We heard many vivid stories about different events and incidents that occurred during the course of the project. The stories included the timing and order of events and opinions of causal connections between

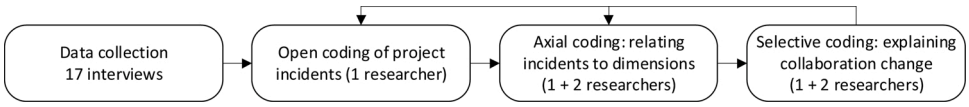


Fig. 1. Research process.

the circumstances, the events, and their consequences. This information enabled us to form a triangulated and combined narrative of the project and the changes in the collaboration that occurred during the development and maintenance of the enterprise system.

In total, we interviewed 17 individuals from Factory, the user organization; integrator, the developer organization; and Middleware Consulting. The interviews lasted between 26 and 73 min, the average being 45 min. Table 1 lists the interviewees, their roles, their organizations, and their temporal project responsibilities. However, because of the long timespan of the project, which spans from 1994 onwards, most of the interviewee's roles and responsibilities had evolved over time. Some individuals were intensively involved in the early implementation, whereas others only had experience working with the recent system changes. Some of the interviewees were no longer at the company.

Table 1
The interviewees.

Interviewee	Role	Organization	Responsibility in the project
F1	Corporate IT manager	Factory	Project leader, 1995–2000
F2	IT manager of a business area	Factory	Project and maintenance management responsibilities since 1995
F3	Program manager	Factory	Business analyst, 1995–2002; business analyst in an interfacing system, 2002–2010
F4	Enterprise architect	Factory	Responsibilities in interfacing systems, 1999–2010; development and maintenance responsibilities, 2010–present
F5	Representative of sales	Factory	Analyst at Integrator, 1995–1997; area responsibilities since 1997
F6	IT support manager	Factory	Area responsibilities since 1995; rollout responsibilities
F7	Representative of logistics	Factory	Area responsibilities, 1995–2006; now working with a related logistics provider
F8	Corporate IT manager	Factory	Project leader, 2000–2004, interfacing systems responsibilities, 2004–2009
I1	Vice president	Integrator	Birdie project manager at Integrator, 1999–2005
I2	Service owner	Integrator	Technical support, 2001–2009, maintenance service responsibilities, 2009–present
I3	Continuous service manager	Integrator	Analyst, 2004–2011, maintenance manager, 2011–present
I4	Infrastructure manager	Integrator	Hardware and facility responsibilities, 1995–2001
I5	Project manager	Integrator	Birdie project manager at Integrator, 1995–1997
I6	Lean software developer	Integrator	Birdie developer since 1999
I7	Service manager	Integrator	India-based offshore maintenance manager, 2011–present
M1	Middleware manager	Middleware Consulting	Middleware Consultant, 1996–1998
M2	Technical consultant	Middleware Consulting	Middleware Consultant, 1996–1998

For this research, it was not possible to access the project archives, as they contained highly confidential material about sensitive business decisions. However, in the 1990s, the first author of this study worked as a developer at Integrator and saw the early phases of the project (1994–1997) from a very close distance. He had management and development responsibilities at Integrator and had experience with the same technologies and development style in an adjacent application area; in his role, he experienced similar crises and their consequences as those we examine in this project. The first author stayed informed on the project and the companies under investigation for the 20 years that followed through various professional and research activities. His knowledge and experience made it easier for us to understand and interpret the interview data. In addition, the interpretations contained in the project narrative were confirmed by two managers who have had major responsibilities concerning the system since the inception of its development.

Our study is longitudinal. Although the data were collected during a limited timeframe, our study approaches time as a social construction whereby "what is critical is not just events, but the underlying logics that give events meaning and significance" ([58], p.273). Although the interviewees from Factory and Integrator emphasized different viewpoints, their views on actual events, along with their descriptions of the reasons for and effects of those events, converged well. Naturally, Integrator emphasized more development practices, problems, and solutions, whereas Factory highlighted business operations and internal issues at the user sites. However, the companies' long, shared history, which spans from the 1960s to the present, contribute to a common understanding of these events.

2.4. Open coding

All the interviews were audio-recorded and later transcribed for analysis. Atlas.ti was chosen as the coding tool, as it provides easy-to-use functionalities for managing text and attaching codes to portions of text. The open coding, performed by the first author, started with a closer scrutiny of the incidents that occurred during the enterprise systems development project. The first author coded the particulars of each incident fully inductively, without an a priori analysis framework, as required by the grounded theory [54,57].

At the beginning, we tried to understand how decisions were formed and why actions were taken in relation to the incidents. However, after coding three or four interviews, a coding scheme started to emerge. This included conditions, decisions, and individual events related to each incident, in combination with the interviewees' presumptions and the effects of the incidents.

Fig. 2 shows an example of open coding, using a corporate IT manager's explanation of how the project became more cooperative after an architecture crisis occurred early in the project. In total, the open coding produced 200 codes, which were classified as events, conditions, decisions, presumptions, and effects (Fig. 3).

2.5. Axial coding

In the axial coding phase, the relationships between the codes and categories were identified. Moreover, new categories based on those relationships were formed. A partial excerpt of a network diagram depicting some of these relationships is presented in Fig. 2. The figure illustrates how the axial coding began. Each project event and decision

A central element was that we could not continue so that we have two projects, that the customer has its own and the supplier their own and they have their own agendas. Instead it is based on a common project, so I established the steering group and continued as its leader and continued as the leader of the whole and then N.N. was more looking at the details. It worked very well, it was so big project that there were levels, like strategic communication and explanation and then we need other people there, where the details and testing and design and so on are. Originally we divided it to two parts. But then I sold the strategy and we established again the project and then it began to work. Even today if I would do a project like that, I would build it cooperatively. There are same elements in the current nuclear power projects, as what I have heard.

Condition: Separate agendas between customer and supplier
Event: change in project organization
Event: create a joint project
Effect: make the project cooperative
Presumption: Clear roles are not possible
Presumption: Tight customer-supplier separation is harmful

Fig. 2. Example of open coding.

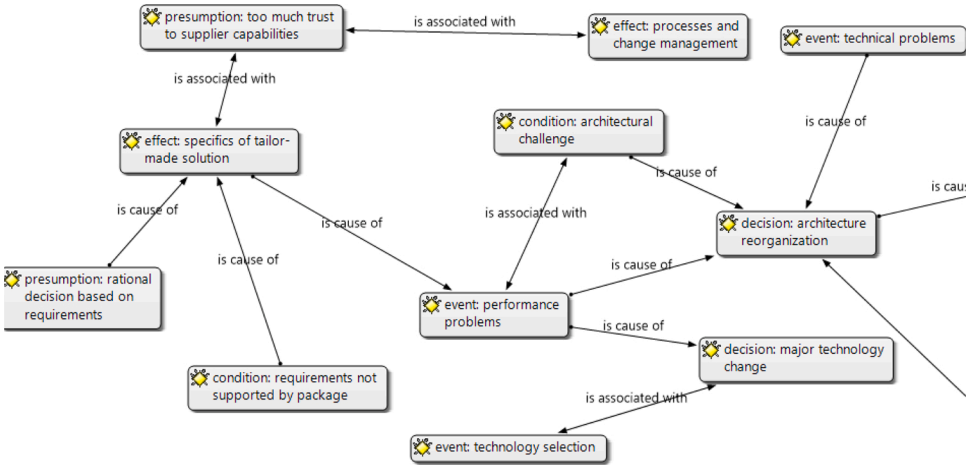


Fig. 3. A network diagram in axial coding.

was scrutinized, and all the related codes were connected to each other. For example, specific issues in the tailor-made system created performance problems, which, in turn, required a decision to be made regarding the architecture reorganization. These relationships were created by comparing items in the dataset, examining how they were related, and determining whether there were any causal relationships between the items.

2.6. Selective coding

The codes related to project incidents illustrate an initial set of dimensions that describe variations during the course of the project history. Each and every incident varied along those initial dimensions:

- Processuality—were the incidents seen as having been resolved by personal contribution or by established processes?
- Cooperation—was the development driven by a deliberate will to cooperate or by legal and commercial contracts?
- Process maturity—were development actions ad hoc by nature, or were they planned and controlled?
- Customization—was the development driven by the requirements or by the package to be installed?
- Customer–supplier—who drove the development during and after the incidents, the customer or the supplier?

We were able to identify changes on these dimensions during and after each project incident. This notion steered our interest toward the collaboration between the main partners of the enterprise systems development project. How the collaboration evolved as the project incidents occurred seemed important. The project, described in detail below, was complex and involved many crises related to the coded incidents. By looking more closely at these dimensions, we can explain the crises and incidents in relation to changes in the collaboration.

Our approach to selective coding can be described as theorizing from process data with grounded theory strategies [59]. Instead of using a priori process theories and testing them with time series or event histories, we delved into the experiences of the actual process participants and formed our grounded theoretical understanding of the events, attaching patterns and meanings to those events and incidents. We selected "changes in collaboration" as the core category and started to refine its meaning and occurrence with the initial dimensions as described above. Soon, we understood that collaboration is dynamic: it does not stay similar over time but changes in response to incidents and varying project needs. We continued the analysis of how and why collaboration changed during the course of this project and what kind of collaboration patterns prevailed after the changes occurred. We looked closer at the dimensions described above and associated them with project incidents. Two of the dimensions, processuality and cooperation, were particularly essential in this theorizing process. Selective coding, by associating dimensions and project incidents, resulted in conceptualization [60] that included four modes of collaboration and four related propositions.

3. Project incidents

A project incident is a major event that occurs during a project, such as a crisis or an internal or external event that changes practices, conditions, or relationships; a technology change; an architecture change; a project organization change; or a major requirement change. The incidents and their consequences were studied in a long-term enterprise systems project that lasted 20 years, from 1994 to 2014.

In the analysis, we identified the major incidents in the development process of Birdie, from the investment decision to its current use. These incidents are then used to describe how collaboration between the

parties involved in the development of Birdie evolved during its life cycle.

The most important incidents related to the development of Birdie are listed in Table 2 and described in detail below.

3.1. Decision to build a custom system (1995)

The development of Birdie started at the beginning of the 1990s with an initial study of the requirements and how packaged ERP systems could support the company. The conclusion was that no existing ERP package could support the desired functionality and business processes well enough, because of the domain and its complex value chain. Although the main products of Factory were bulk raw materials sold to other businesses, the production items had unique identities that needed to be tracked over their whole lifetime for logistics and quality assurance.

As there was no packaged ERP with a suitable conceptual model of bulk titles and unique items in the 1990s, Factory decided to build a unique system fully tailored to its needs. It was evident that Integrator would deliver the system. A part of Integrator originated from Factory's IT department. The company had also built many of the operative legacy systems that the new system would replace. Thus, there were already established collaboration practices, domain knowledge, and close personal relationships between Factory and Integrator at many levels, which had developed through the development and maintenance of many individual systems. Despite these close and personal relationships, the decision to build Birdie as a requirements-based, tailored system was seen as a rational decision, which was made after careful research and a weighing of the alternatives, including an evaluation of SAP and other packaged ERPs.

Table 2
Main project incidents.

Year	Incident	Description
1995	Decision to build a custom system	Factory compares packaged ERPs with its requirements and decides to build a custom system.
1995	Decision to build a general product for the industry field	Integrator aspires to reach new markets with the decision to build a general product that is based on Birdie.
1996–1998	Project start and technological crisis	Full-scale development starts. Very soon, the project encounters a crisis related to the selected technological approach.
1998	Architecture reorganization	The technological crisis is resolved by cooperation between Factory and Middleware Consultants.
1998	Merger	Factory merges with a competitor of about the same size.
2000–2004	Rollouts	Birdie is installed and taken into use globally in approximately 50 production sites and sales offices.
2000	Abandonment of product development	Integrator concludes that it is not possible to make a general product of Birdie because of its very Factory-specific features.
2000–2002	Move to maintenance	New development is gradually replaced by a more maintenance-oriented development.
2002, 2008	Change in technology	The server-operating systems and the database management system are changed for commercial reasons.
2006, 2010	Offshoring	Maintenance and further development are offshored first to Europe and later to India for cost reasons.

3.2. Integrator aspires to build a general product for the industry (1995)

The business domain where Factory operated had strategic importance for Integrator. Integrator hoped to achieve a global presence in this domain. Hence, when Factory concluded that no existing packaged ERP fulfilled its needs, Integrator saw an opportunity to build a general product based on Birdie that could be offered to other global enterprises in that business domain.

The decision to use Birdie as a basis for a general enterprise systems product increased the complexity of the project and created hidden motives and agendas in the interactions between Factory and Integrator. While the development was previously based upon common practices, good will, and personal relationships, now, an opaque component entered whereby some stakeholders speculated about future benefits from the competitors of Factory. This was particularly evident in the selection of development tools and libraries. It was perceived as important that no license fees for tools and libraries were included, and that these elements were selected for long-term product development with portability, and not for rapid and flexible customer-specific implementation.

3.3. Project start and technological crisis (1996–1998)

Despite careful consideration, the project started with a technological crisis. The requirements and business processes were reasonably well understood by both Factory and Integrator, but the selected technology was unstable. Software and systems development in the 1990s can be characterized by immature technologies, constant changes in technology, and new development tools and platforms. This was also the case for Birdie. It was originally developed as a POSIX compliant C++ application for Windows NT clients, using UNIX servers and a relational database that replicated over data communications networks that were, at the time, unreliable. The platform included a homemade class library for fat clients and a proprietary messaging service that was planned to be used globally for business transactions. The messaging service that was used for delivering transactions between sites was performing well, but the client class library was still immature in its development when the implementation of Birdie began. In addition, most of the developers were not familiar with the platform. Their earlier experience was mostly in character-based UNIX and MPE systems. These challenges accumulated, causing slowness at the start and confusing the developers.

The first steps in the implementation of Birdie were not easy. The biggest obstacles were poor performance and inadequate scalability, both of which were a result of the wrong architectural choices. The first deployment of Birdie was scheduled in 1997. It soon became clear that this would not happen. The selected architecture for fat clients and the extensive interactions with database servers could not produce the required performance for real-life situations. The client library generated too much interaction with the database servers because of a fundamental flaw in the architecture design. This caused insurmountable performance problems. In addition to being a severe business crisis for Integrator, the awareness of crisis grew both at Integrator and at Factory. The project costs started to grow, and the schedules began to slip. It became clear that it would not be possible to replace the legacy systems before 2000. At this point, Factory understood that the contract it made with Integrator was overly optimistic and would not substantiate as such.

3.4. Architecture reorganization (1998)

Factory's proactive problem-solving activities were essential to overcome the crisis. Factory's project manager took the lead, and through his personal connections, he hired external experts from Middleware Consultants to redesign the architecture of Birdie. This produced another crisis at Integrator, as many of the key technology experts felt that their expertise was ignored. They decided to resign.

This incident and the recovery process can be characterized as a phase of improvisation, intensive cooperation, and personal contribution. Many of the interviewees told stories about this phase: the crisis was resolved through personal contribution and cooperation in reorganizing the code and testing the new architecture in real situations. Factory also understood that to overcome the crisis, it must provide some new benefits to Integrator. The compensation described in the original contract was not viable for Integrator from a business perspective. This understanding, along with a very good business cycle in the field, made it possible to emphasize collaboration in the project.

3.5. Merger (1998)

While Birdie was gradually getting back on track, new, unexpected external events created concerns. Factory decided to merge with an international company of about the same size. This was a source of much uncertainty and had to be handled by the project managers. For example, after the merger, it was not clear which system—Birdie or its counterpart at the other company—would be selected for the new, merged Factory.

The headquarters of the other company was located in another country, which created additional pressures. Many interviewees mentioned that there was a feeling of competition between the countries. The employees in both countries wanted to have a say in the system. As Birdie was at the very core of the Factory value chain, it was of utmost importance for the business units to have a working and usable system.

The process of selecting the system to handle the core value chain of the new Factory was not straightforward. This process also included a political component, as both of the former organizations in the new Factory wanted to have a system that would be beneficial for their individual interests. In this discussion, Birdie was saved by the fact that there was no evident and ready-to-use alternative. It also received some unexpected external support from another large enterprise, which selected the same Middleware solution simultaneously, despite the fact that some believed it would soon become an obsolete technology. Factory decided to deploy Birdie at its core production sites globally in 2001. In 2004, the decision was made to deploy Birdie at all possible sites.

All of the above overlapped with the architecture reorganization of Birdie, which means that Birdie and its technology were in flux as the business environment made tectonic moves.

3.6. Rollouts (2000–2004)

After all the crises and uncertain periods, Birdie was successfully deployed to manage Factory's core value chain, with observed benefits. By the end of 2004, Birdie had been installed and deployed at 33 European sites. The number of deployments meant that Factory and Integrator together needed to create explicit processes for testing, change management, and rollout implementation. Because of the large number of sites, the rollouts were done in parallel. This generated additional needs for defined processes.

In the spirit of collaboration, the rollouts were managed and organized by Factory, and the implementations were supported by experts from Integrator. At the time, Factory had hired a new project manager for Birdie. One of the first things the new manager observed was the lack of change and quality management processes. There was a clear need to "professionalize" recurring activities in the implementation. ITIL [61] was seen as a solution for moving toward professional change and quality management practices.

3.7. Integrator abandons product development (~2000)

Roughly in 2000 (the exact timing cannot be determined from the data), Integrator concluded that Birdie would be a unique solution and would not yield any new product opportunities. The data suggest that this decision influenced processes and attitudes at Integrator. There was less need to focus on IPRs; the goals were less diverse, and customer-specific processes were easier to promote.

3.8. Move toward maintenance (2000–2002)

In 2000, an official decision was made to end the development project and move Birdie to a maintenance organization. However, major development efforts were ongoing until 2002. As releasing new versions, bug fixes, and features to a very large install base was difficult and complex, ITIL was gradually taken once again as the basis for process development, now for the development of maintenance processes.

3.9. Technology changes (2002, 2008)

In 2002 and 2008, the tight connections to certain technology providers were cut. The key technologies in Birdie's application and database servers were changed, and both changes were made for cost reasons. In particular, it emerged from the data several times that the database technology was changed after Factory felt that the database provider became greedy and wanted to increase profit margins. No major difficulties with regard to either change were emphasized in the data. However, both changes were moves toward a more impersonal direction with regard to technology. In both cases, the new technologies were standard products offered by Microsoft.

3.10. Offshoring (2006, 2010)

The costs were also a determining factor when Factory decided to offshore the maintenance of Birdie. They signed a contract with Integrator to nearshore the maintenance to a cheaper country in Europe in 2006 and to later offshore it to India in 2010. Again, both changes entailed new requirements for the processes of managing changes and testing its functionality. Nearshoring and offshoring also moved the maintenance organization toward a more impersonal direction. The old practice, which involved maintaining the personal relationships between Factory and Integrator, was no longer possible—at least not to the same extent as before.

3.11. Epilogue

At the time of the interviews (2014), Birdie was widely used at Factory. It was also considered a success, despite the initial problems. Birdie seemed to serve the core value chain of Factory well, probably better than any packaged ERP ever could have. Yet, most of the interviewees argued that such tailored development would not be possible any more. Doing so would go against the current trends in systems development and IT management. A roadmap for the future of Birdie was under construction, and it was unclear which direction the development would take. Some interviewees thought it was probable that Birdie would continue to run for at least five years from the time of the interviews.

4. Analysis and findings

4.1. Interorganizational collaboration modes

We examined the incidents in Birdie's development history under closer scrutiny and used open coding to identify the specifics of each incident. First, we wanted to see how the incidents emerged and what kind of decisions and actions the user organization took in reaction to them. We wanted to understand how much of the decision-making was rational and how much involved reacting to external and internal events and the arbitrary conditions of the moment. In this investigation, we used a coding scheme whereby we identified the conditions, the

decisions, and the individual events related to each incident.

Upon closer inspection, it became clear that most incidents caused major changes in the collaboration between the main parties, Factory and Integrator. The project incidents, along with their resulting changes and crises, caused breakdowns in collaboration, forcing changes to the collaboration mode. We reconsidered these incidents and changes to be breakdowns [62], as we were not able to explain them easily using existing theories such as Gregory et al. [38] control configurations. It therefore seemed critical to understand what happened to the collaboration during and after the incidents. The decisions and actions taken were not entirely reactive and random, but they were related to emerging awareness of the different collaboration possibilities caused by the incidents.

Using axial coding, we identified patterns in the changes in collaboration after and during the identified project incidents. In later steps of the analysis, we explain why and how the collaboration between the main parties changed during and after the incidents as well as what determined the direction of the change.

Using selective coding, the entire dataset was reinterpreted from the perspective of changes in collaboration. Important categories that emerged included contracts, cooperation, personification, and processes. The more we investigated the data, the more plausible it seemed that there were four different and independent "modes" through which collaboration took place:

- **Contractual mode**: Collaboration is defined by legal contracts between the parties. The project incidents and their consequences are handled according to formal business contracts that define the roles and responsibilities of the parties.
- **Cooperative mode**: Collaboration is based on mutual interests and voluntary cooperation. The parties observe the answers to the incidents and their consequences from their common points of interest, so that the solution is beneficial to both parties.
- **Personified mode**: Collaboration happens between individuals. The incidents and their consequences are dealt with by the key persons who may then delegate responsibilities in their respective organizations. The relationships between the key persons may have developed over years of collaboration. The key persons acknowledge each other's expertise and recognize the areas of trust.
- **Process mode**: Collaboration is a process that can be planned and designed. The incidents and their consequences are resolved through a defined process that determines the parties and procedures. Typical defined processes include those related to change and quality management.

The following quotation exemplifies the contractual mode in 1996–1998. It describes the collaboration mode during a project crisis, when there was growing tension at both Factory and Integrator and a desire to toss out the complicated, rigid, and restrictive contract and instead adopt the cooperative mode. Factory understood that following the contract strictly would not lead to any success. Instead, a more flexible and equal collaboration mode, accompanied by control agreements, was needed:

"I think that was the most challenging part: to give up the initial mindset of 'you have the stuff, and we have the money.' You'll deliver the stuff, and we'll pay you. We have this agreement, which juristically binds us to do things. You just have to obey it." (Corporate IT Manager, Factory).

During the development of Birdie, the incentive to cooperate materialized during the technological crisis in 1996–1998. After unsuccessful attempts to resolve the problems, both organizations realized that they had the common objective to rescue the project:

"Let us say this. Usually, projects are saved by the fact that the customer and the vendor are equally deep in the [rude expression removed]. Then, there is a willingness to proceed and get the thing sorted out." (Middleware Consultant)

At Birdie, the cooperation mode extended so far that the project organizations at Factory and Integrator were de facto merged without an explicit renegotiation of the contract. The customer took the lead and reorganized the project:

"The main element was that we couldn't continue as earlier. There were two separate projects: the customer had one, and the vendor had another one, each with their own agendas, etc. So, I decided to establish a joint project. I set its steering group, and I continued as its chairman, in charge of this whole thing and overseeing everything." (Corporate IT Manager, Factory)

Adopting the personified mode means that heroic individual accomplishments and the importance of individual expertise are emphasized. This occurred when the technological crisis had to be resolved. The customer-side project manager made an alliance with the supplier's infrastructure manager and invited an external consultant to resolve the Middleware problem:

"I had this personal relationship. I realized that we're going to ride for a fall. So, I invited that guy here. He flew over on a morning flight, we sat in [the local restaurant], and I told him everything. I had all the documents, and I explained which is which. Then, we had lunch. After this, he said that we are really in trouble, but he is going to help us out." (Corporate IT Manager, Factory)

"But we had some common history. Me and [Project Manager, Factory] had worked together [on an earlier project]. There, we faced these issues on a smaller scale. He managed the project. I was the infra provider, a kind of safety. And we applied these experiences, and I argue that it was quite useful for both of us." (Infrastructure Manager, Integrator)

Later, when the crisis was resolved, the importance of recurring processes grew. The process mode was adopted in change and quality management. However, the switch from the cooperative mode to the process mode was not simple. Instead of focusing on the essential and continuous process of managing change and quality, the project organization had concentrated on resolving the immediate and fundamental development problems at hand. No plans, models, or processes were established for managing the frequently recurring actions in change and quality management:

"When I came in, I thought it was chaos. Like I said, nobody knew how many change requests there were, what kind, and where they were. They were nowhere; they were in different places. Then, we made it systematic. We established the whole testing model, the whole change management, how to make new releases, how many weeks can we use [Integrator] and where, how much they do, where the acceptance criteria are, and how many changes we may accommodate. If there are acute changes, when can they come, the last 20 percent. When each person tests it, and then we could develop the testing process as well. In the beginning, it felt that the stuff from [Integrator] hadn't been tested at all." (Project Manager 2, Factory)

4.2. Incidents and their effect on the collaboration mode

Again, we went through the data and analyzed what promoted and demoted a certain collaboration mode in the context of a project incident. This analysis is summarized in Table 3. The "+" sign in a cell indicates that the item promoted the collaboration mode, and the "-" indicates that the item demoted the collaboration mode. Because of the extensive space required, it is practically impossible to display all the evidence related to the items in the table. We do, however, illustrate the evidence with a detailed example of one incident, "Project start and technological crisis."

At the beginning of the project, ingredients from each collaboration mode were present. A contract was made between the parties according to the specified requirements. A user-oriented approach promoted the cooperative and personified modes, and the established ways of working between Factory and Integrator promoted the cooperative and process modes. Factory's do-it-yourself culture probably also promoted the

Table 3
Incidents and their effect on collaboration mode.

Incident	Contractual mode	Cooperative mode	Personified mode	Process mode
Decision to build a custom system	+ requirements - based contract	+ user-drivenness + established way of working	+ user-drivenness + do-it-yourself attitude	+ established way of working
Product development decision	+ secure supplier's legal position	- supplier's hidden objectives	+ technology and product experts	
Project start and technological crisis	+ awareness of over-trust in the supplier	+ awareness of crisis	+ solutions through personal relationships	- project reorganization
	- unrealistic contracts	+ customer-driven technology selection + supplier's business crisis	+ ad hoc crisis management	- ad hoc crisis management
Architecture reorganization	+ awareness of over-trust in the supplier	+ compensation for supplier losses	+ expertise through personal relationships	- trial-and-error approach
	+ use of external problem-solvers	+ customer-driven problem-solving	+ opinion that "we need world-class expertise"	- unrealistic developer beliefs
	+ competition among suppliers	- competition among suppliers	+ management's risk-taking confidence - resignation of key developers	- awareness of crisis
	+ responsibilities of merged IT	- competing systems	+ clear decision points with competing parties	- decisive events with contingent results
Merger	+ competition among suppliers + use of consultants to confirm decisions	- fractions and parties - cultural differences	- more stakeholders - increase in the scale of the system	+ more stakeholders + awareness of process needs
	+ more external parties	- location and ownership issues		+ added requirements and changes in needs
Rollouts		+ customer-driven process		+ clear management support + recurring activity
Abandoning product development	- less need to protect IPRs	+ less-diverse goals + easier to fulfill customer requirements		+ customer-specific processes
Move to maintenance	+ use of standards	- business as usual	- complex releases - very large install-base - testing requirements	+ change management + use of standards + complex releases + very large install-base + testing requirements
Changes in technologies	+ cost control for licenses		- more generic attitude to expertise - partner greediness	
Offshoring	+ focus on costs + scale-down + personnel changes	- focus on costs - personnel changes	- added distance - personnel changes	+ offshoring requires clear processes + personnel changes

personified mode. Integrator's decision to build a product was a move toward the contractual mode. This move introduced new legal issues and hidden objectives that demoted cooperation. It also elevated the role of certain experts at Integrator.

The technological crisis was solved partly by moving from the contractual mode to the cooperative mode. Despite this move, there was a factor that promoted the contractual mode as well. Factory realized that it had trusted Integrator's capabilities too much (awareness of over-trust in the supplier):

"We were a little amateurish. I guess at Factory, we trusted too much that Integrator knew what they are doing. But they didn't. They just continued on the same basis as before but didn't confirm the functioning. So, this was maybe the most amateurish mistake from the very beginning of the project." (IT Manager, Integrator)

However, the awareness that the contracts were unrealistic demoted the original contractual mode (unrealistic contracts):

"The objective was for it to become a kind of customer-supplier project, and we ordered everything with invitations to tender, and so – so we started to do it." (Corporate IT Manager, Integrator)

This contract principle did not work. The increasing awareness of the crisis and Integrator's business crisis had a decisive effect on the collaboration mode. It promoted the cooperative mode:

"Then, it really hit the roof. I got an invitation to Integrator's meeting. There was their whole management team. Then, project management, and then Integrator's CEO said plainly that if this does not work, the company will be bankrupt." (Middleware Consultant)

"Maybe it was partly that—partly the risks realized—and it was about the ambitions. It was the biggest system that they had ever made. At the same time, they took this object-oriented approach. The skills and expertise risks realized, and, in a way, they thought too much of themselves, their skills and their experience." (Middleware Consultant)

The crisis strengthened the role of Factory, the customer, also in technical details (customer-driven technology selection). We interpreted that this stronger role on the customer's part promoted cooperation.

"[Factory Project Manager] then managed to also persuade Integrator to support this, even though this was a huge change. It took a year for Integrator to recode the two-level architecture to the new three-level architecture, adding Tuxedo in-between." (IT Manager, Integrator)

The crisis also required improvisation and ad hoc actions (ad hoc crisis management). We interpreted this as a move to the personified mode.

"Then, [Factory Project Manager] came to our offices one Monday night. He tried to get us to understand that this would totally fail. Then, I jumped up and said, 'Yes, now I understand.' We got lots of internal hassle, but I called my boss at eight in the evening and said that now I need half a million euros. For what, he asked, and I said that now we are going to buy the biggest servers possible and that they will be flown here from California; otherwise, we will be doomed. He listened to me and said, 'Okay.' I called [hardware vendor's] sales director at home and said, 'Call California, and tell them to put six power servers on the plane immediately,' and so it went." (Infrastructure Manager, Integrator)

Moreover, personal relationships played an important role in resolving the crisis (solutions through personal relationships):

"We had a seminar at [location], and [Factory Manager] was among the participants. I said then that indeed, this is interesting, and then [Factory Manager] came in the first row. There were about 150 participants, and he asked me to have a chat and said they have a little challenge at Birdie, and could we come and help? I said, 'Yeah, I have

been waiting for this.' [Factory Manager] came to the project as a fresh manager, which was surprising to me. I had known [Factory Manager] for a long time." (Middleware Consultant)

As in many crises, the processes were neglected, and the project organization was reconstructed (project reorganization), which demoted the process mode, at least at the beginning.

"I think it was a very cohesive project team. And then, we had no possibilities for virtual interaction as we now have. It became a community. And we widely used the capabilities and skills of various parties." (IT Support Manager, Factory)

The architecture reorganization meant returning to the contractual mode, although cooperation and personification were still considered important. Factory understood that it had trusted Integrator's capabilities blindly, so it had to build a new contractual basis for the project. A new kind of compensation for Integrator's losses was embedded in the contract. This, in addition to the contractual mode, enabled and supported the cooperative mode. The use of external problem solvers, the Middleware Consultants, emphasized the contractual basis. These data also indicate that there was some competition among the suppliers, which may have demoted cooperation. There was also an opinion that there was a need for "world-class expertise" to solve the architecture problem. This need was met by the personal efforts and cooperation between certain key persons, such as Factory's project manager, Integrator's infrastructure manager, and the Middleware Consultants. The parallel resignation of Integrator's key developers was a step toward more impersonal responsibilities. As the project was still in a state of crisis, a trial and error approach and personal efforts prevailed. This clearly prevented the relevance of the process mode.

Although the merger was an enormously important event for Factory, it did not produce immediate changes to the project collaboration. Perhaps because of the very recent crisis and the architectural challenge, the project collaborators continued to work in the cooperative mode. From the long-term perspective, the merger brought along pressure to move from the cooperative mode to the contractual mode and the process mode. An increase in the number of stakeholders and in the system's scale required more impersonal, defined processes. In addition, the merger brought new requirements and essential changes to Birdie. These, in turn, required better management processes. However, after the technical architecture challenges were resolved, most of the decisions concerning Birdie were political. For example, political struggles between the parties at Factor's different locations required additional consideration:

"Then they would have taken a similar system into use from [another country of Factory]. Birdie ran them over; they did not succeed. They had quite massive systems. It's quite a political struggle—which system they use at big companies. Each system has its own proponents. It's a tough struggle, and they tried to undermine Birdie from [another country], until based on the [consultant] statements and everything, Birdie moved them aside." (Project Manager, Integrator)

The political fights required committed and politically skilled persons. We heard stories about situations where Birdie was in deep trouble. We interpreted these situations and their results as quite contingent: the decision concerning Birdie could also have been negative.

The rollouts promoted the process mode. The rollouts were a recurring activity (44 in total), requiring well-defined processes for different teams. Explicitly articulated management commitment also promoted these processes. Although Factory led the rollouts, the process still required intensive cooperation between Factory and Integrator. Our interpretation is that Integrator's decision to abandon product development promoted the cooperative mode, as it created better opportunities to build customer-specific processes for Factory.

The move to maintenance also promoted the process mode. Change management, testing, and releasing required clear processes, which were taken from the ITIL standard. As the project was no longer in crisis mode, maintenance was seen as business as usual. The situation promoted the contractual mode, while demoting the cooperative and personified modes. Instead of being based on cooperation and personal relationships, the development was now becoming an impersonal process based on the business contract.

The move to maintenance was the only project incident that included technical features that directly affected the collaboration mode. The technical features (e.g., complex releases, a very large install base, and testing requirements) (see Table 3) required well-defined processes and improved management. All the other items in the other areas of Table 3 related either to the social, organizational, or business context. While the items may include a technical component, the main effects and consequences are not based primarily on the technical qualities.

While it may sound like the changes in technologies were a technical decision, they mostly resulted from a business decision: a reaction to increasing license costs. It was also a transition out of the personified mode. The companies selected Microsoft technologies, a more generic choice that did not require direct personal connections to the technology provider. Offshoring maintenance was yet another step toward the process mode. It removed the personal relationships between Factory and Integrator that had been present and required new maintenance processes and a new contract between Factory and Integrator, which thus changed control of the project.

5. Discussion

5.1. A mid-range theory of four collaboration modes

The history of Birdie can be explained by the alteration of the four collaboration modes: the contract mode, the cooperative mode, the personified mode, and the process mode. We believe that understanding these high-level collaboration modes is as important as understanding more concrete artifacts [42–44] in boundary-spanning cooperation. Each collaboration mode was emphasized differently during and after every project incident. Table 4 distils the theoretical model derived from this study and characterizes the collaboration modes by defining their essential features. We consider this the parsimonious presentation of an inductively created middle-range theory that explains the collaboration modes and their changes in the context of large enterprise systems development projects.

This middle-range theory can be described as follows. A specific collaboration mode is a reaction to a condition or incident—or more likely, a chain of incidents. The contract mode is a reaction to the need to define the division of costs and responsibilities as early and as clearly as possible. This is similar to outsourced development arrangements and can be traced back to transaction cost economics [63]. It can thus be argued that the contract mode is typically seen as the mode of choice

Table 4
Collaboration modes and their features.

	Is reaction to, solves the problem of	Regularity	Emphasizes	Requires
Contract mode	Division of costs and responsibilities	Management-induced system development	Plans and commitments	Clear contracts
Cooperative mode	Lack of clarity in the context	Cooperative development of new solutions	Cooperative action, the spirit of "us"	Common goals
Personified mode	Imminent problem-solving needs	Improvised problem-solving actions	Individual achievements	Influential persons
Process mode	Constant planning needs	Planned development actions	Change and quality management practices	Defined and implemented processes

when starting projects because it transfers much of the risk from the buyer to the seller. The contract mode is suitable when there are very clear requirements, the work can be divided effectively, and the parts of development can be isolated. The cooperative mode is adopted when the context is unclear, such as in the case of Birdie's technological crisis. The cooperative mode is forced by difficulties in dividing responsibilities in response to an acute crisis or a problem that spans functionalities and responsibilities. The personified mode is often a reaction to imminent problem-solving needs that require improvisation and ad hoc actions; this mode can work very well when there are acute crises that can be pinpointed into a single origin (e.g., the actions of an architecture guru or a senior decision-maker). Any long-lasting "normal" situation that emphasizes planning needs promotes the process mode; the process mode is effective in the context of continuous development and both routine and planned maintenance after major development efforts. In the modern development context, the process mode can be implemented through continuous practices and principles such as those labeled as DevOps [64].

Each mode emphasizes different elements of systems development. The contract mode sees systems development as a management-induced action, emphasizing plans and commitment from the management. In the cooperative mode, extensive cooperation is required to realize new solutions [65]; the cooperative mode emphasizes the spirit of "us" in cooperative action. In the personified mode, improvised problem-solving actions are normal; this mode clearly emphasizes individual achievements. The process mode is business as usual, whereby planned development actions are executed; during the development of Birdie, change and quality management [66] were emphasized in relation to the process mode.

Each collaboration mode has also different prerequisites; if these are present, the mode is more likely to be a good solution to the encountered problems. The contract mode requires clear contracts, clear requirements, and an effective division of work for the isolated parts of the project. It is not possible to adopt the cooperative mode without the identification of common goals [67], even if that goal is simply to resolve common problems and difficulties; this can create a dynamic and extended mode of cooperation that goes beyond what is specified in the contract. The personified mode requires influential persons who are committed and able to use their influence, skills, and personal relationships; in this case, these individuals created their own networks across organizations and solved problems with their influence. For the process mode, defined and implemented processes are required, particularly in the areas of change and quality management; in this case, these processes were needed and established when the system was installed and adopted by multiple global sites. We believe that similar things happen in most projects near deployment.

We can distill this analytic generalization in the form of propositions. These propositions are not mutually exclusive. The modes can be mixed or concurrent and occur in different parts of organizations and work subsystems. The four propositions are:

- P1 When the focus of interest is on the division of costs and responsibilities, the contract mode is emphasized.
- P2 When there is a lack of clarity in the development context and common goals among partners can be identified, cooperative mode is probable.
- P3 When there are very imminent problem-solving needs (i.e., a crisis), influential persons take the lead and the personified mode appears.
- P4 When there are constant planning needs and regular planned development actions, the development moves to process mode.

Fig. 4 shows examples of how these propositions occurred in the project, i.e., how collaboration modes were emphasized in and after

Project incidents

Year	Incident	Description
1995	Decision to build a custom system	Factory compares packaged ERPs with its requirements and decides to build a custom system.
1995	Decision to build a general product for the industry field	Integrator aspires new markets by a decision to build a general product that is based on Birdie.
1996-1998	Project start and technological crisis	Full-scale development starts. Very soon it reaches a crisis related to the selected technological approach.
1998	Architecture reorganization	The technological crisis is solved by a cooperation of Factory and Middleware consultants.
1998	Merger	Factory merges with its competitor of about the same size.
2000-2004	Rollouts	Birdie is installed and taken into use globally in about 50 production sites and sales offices.
2000	Abandonment of product development	Integrator concludes that it is not possible to make a general product of Birdie because of its very Factory-specific features.
2000-2002	Move to maintenance mode	New development is gradually replaced by a more maintenance-oriented development.
2002, 2008	Change of technology	The server operating systems and the database management system are changed for commercial reasons.
2006, 2010	Offshoring	Maintenance and further development are offshored first to Europe, and later to India for cost reasons.

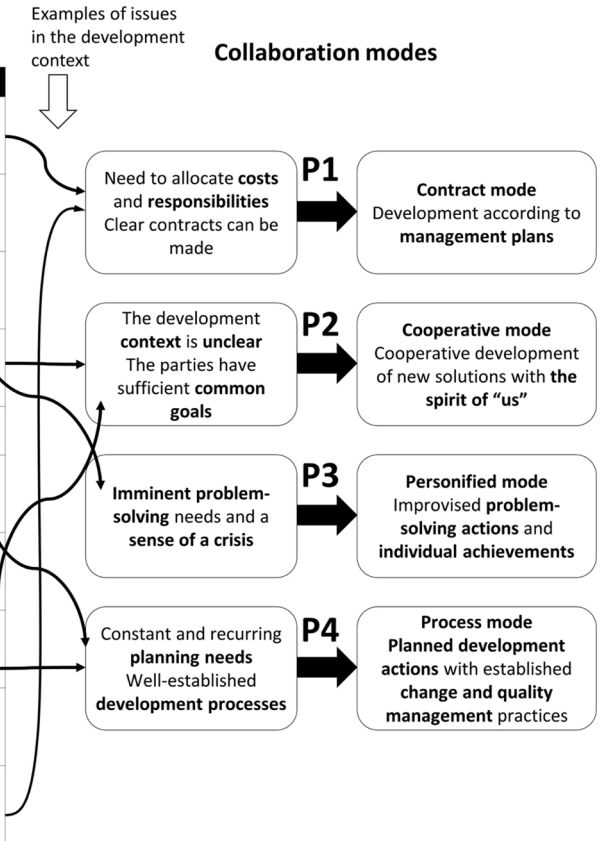


Fig. 4. Project incidents and collaboration modes.

project incidents. Thin arrows show examples of occurrences of a certain condition during the project. Thick arrows (P1−4) show how a certain mode existed and provided grounding to the propositions. The collaboration modes existed simultaneously throughout the development of Birdie, but their relative emphasis varied a lot during the project's history. The observations from the parties at Birdie indicate that during and after each incident, the project organization may start to emphasize another mode. This transition can happen without any deliberate decision, or it can be explicitly decided. A reaction to an incident can be followed by changed awareness with regard to the requirements of the project collaboration. From the history of Birdie, we can explicitly identify shifts between all four collaboration modes.

5.2. Implications for research and practice

We have presented a single case study of a large project with a long and winding history. As such, it is not unique. The individual issues that the project contributors encountered, including a heavy emphasis on contracts as the basis for cooperation [68], overreliance on heroic actions by individual developers [69], and the evolution from reliance on individuals to more impersonal processes [70], are typical for large enterprise system projects and for most ISD projects that span a long period of time and involve several organizations. Our study, however, differs from previous studies (c.f. [35]) in that it observes collaboration change over a long period of time, as is done in the process-theorizing research [3,38]. We also analyzed both how the individuals perceived change and how the change affected the project and the organization. Therefore, we see our approach as a holistic one, with direct evidence and grounding to real-world observation of a specific enterprise systems development case. In this sense, our approach differs from—and enriches—for example, the work of Ring and van de Ven [71] and Juell-Skielse et al. [36], who theorize about interorganizational collaboration in general, and Yeow et al. [44], whose work stresses the role of specific boundary objects and a boundary organization (see also [43,72]). Our collaboration modes resemble Gulati et al.'s (2014) and Juell-Skielse et al. [36] modes, but they see them fixed at the beginning of collaboration and to stay intact for the duration of the project. Furthermore, we offer a richer view of what causes changes in collaboration and collaboration levels as compared to recent control dynamics studies [38,73], as we demonstrate that issues outside of the direct collaboration context (e.g., technology changes or the outsourcing of partner goals) also influence collaboration. We thus respond to Sabherwal [74] call for the study of the processual aspects of interorganizational dialectics through longitudinal studies.

Next, we discuss our results in relation to earlier research and the implications for enterprise system development practices.

5.2.1. Research contribution

We observed changes in collaboration over the course of a long development project. We identified four basic modes of interorganizational collaboration in the development process: contract mode, cooperative mode, personified mode, and process mode. The choice between them (or the need to apply one or more) is a reaction to environmental conditions, which are most often business related. The collaboration modes are not mutually exclusive; however, while they may exist simultaneously, one collaboration mode is usually emphasized at any given time. The selection of a collaboration mode depends on the given situation (Table 4 and Fig. 4). When a certain collaboration mode is emphasized by the project partners, it strongly influences on how the project continues. We do not claim that we have identified the causal relationships between project incidents and collaboration modes. Instead, we describe what kind of problems the project partners attempt to resolve when they adopt certain collaboration modes or when a new collaboration mode emerges.

Different collaboration modes address different problems, emphasize different forms of collaboration, and require different tools, artifacts, and practices. We propose that certain types of issues can be addressed by the movement to certain collaboration mode (Table 4 and propositions in Fig. 4). Previous research has identified partially related constructs, such as control configurations [38], collaboration-related objects [39,50], boundary organizations [44,72], and boundary spanners [75,76] as objects and carriers of cooperation and knowledge across organizational boundaries. We complement these constructs with the four collaboration modes and propositions of when they should be emphasized.

Our distinct contribution is that unlike most past studies, particularly those based on success factor models [24,25,77], we studied the evolution of the relationships between the development parties over a long period. Previous studies have often assumed, either explicitly or implicitly, that rational decision-makers select the presumably optimal collaboration mode in advance and use it throughout the duration of the project, moving through different phases of collaboration but maintaining the same mode [74]. In contrast, we observed that in a prolonged development context, different kinds of changes in the external and internal contexts or different stakeholders and their goals force the parties to renegotiate their deals and to adjust the collaboration mode accordingly. Although it is probably impossible to predict the exact consequences of certain events on the collaboration mode, our findings explain the types of collaboration problems that partners can potentially resolve by adjusting their collaboration mode. This is described in the first column of Table 4.

Tsoukas and Chia [78] claim that organizations are composed of change. They are "sites of continuously changing human action" and "a patterned unfolding of human action." Our findings suggest that this is particularly true in the context of interorganizational systems development, whereby temporary development organization is constantly changing and reacting to internal and external project incidents and their needs. Furthermore, collaboration cannot be easily studied at either the individual or organizational level. Instead, one must understand the changes at both levels, as their relative importance varies over time and according to the development situation. From this perspective, we respond to the calls for studying systems development through process approaches and in real-world settings [51] and to calls for studying processual aspects of coordination evolution [74].

Although a systems development project initiates changes to the organization, the project itself is often seen as a stable entity through which the partners work toward an established goal. Our findings question this view and position the concept of a project as a volatile process [78,79]. Hussenot and Missonier [80] view organizations as structures of events, which is a natural way to understand organizations. It became much easier to interpret our case organization when we started to analyze it as a series of incidents instead of a set of organizational entities. Similarly, Hernes and Weik [81] pre sented a theoretical classification for how organizations can be presented as processes. The classification is based on how structure (i.e., independent entity) is applied to an organization, which they call "entification," or the process of conceptualizing entities in the analysis. They argue that there is no natural split between the process and the entity and that there are different ways to divide them. Instead of viewing the project as a stable environment where information and artifacts flow from entity to entity, our case was blurred and dynamic. To use Hernes and Weik's terminology, we had an endogenous view of the process whereby the process constantly reproduced itself. Our collaboration modes thus show how temporary organizational arrangements are produced. Changes in control, triggered by the changes in the project context, tend to be recursive (c.f. [38]).

Our findings are also in line with Langley et al. [79] view on the importance of process conceptualizations. They posit that there is a need for process approaches that use longitudinal and rich process data, and they approach systems development as a continuously and dynamically changing process that cannot be fully predicted. It is quite easy to draw the analogy from Langley et al.'s call for process theories to recent

changes in software development, whereby agile [82] and continuous approaches [83,84] are the reality in many organizations. In enterprise systems development, agile and continuous approaches are still in their infancy, which calls for future research and empirical observation of these practices.

Previous research has proposed different ways of collaborating, including outsourcing arrangements [68], controlling consultants [1], and forming partnerships [85]. Our work went beyond simply identifying the incidents and collaboration modes by revealing the existence of several collaboration modes and the shifts between these collaboration modes in response to incidents or events. We believe that this mid-range theory can help researchers interpret events that occur within a collaborative project and identify successful practices that can be used to overcome the problems that arise during projects and potentially change the course of the project radically if needed. The results do not provide a recipe for success. However, they are important for any initiative that attempts to improve practices and life cycle models for large-scale enterprise systems development. Our work demonstrates that in all kinds of cooperation arrangements, it is important to monitor not only the performance of the partners [1], but also the overall ways of working; this can inform active decisions about the most suitable collaboration mode. We believe that similar events can occur also in large infrastructure and embedded software projects (e.g., 737 MAX issues in [86]) as they do in enterprise systems development.

5.2.2. Implications for practice

Our theoretical results have direct practical implications. First, they can be applied when considering and planning outsourcing and partnerships. Second, the changes in the collaboration mode over time should be considered in development methods and processes. Third, although predicting changes in the collaboration mode is difficult, the changes can be expected when a development project progresses or when an incident occurs. For example, in most cases, the transition to maintenance yields a shift from the cooperation mode to the contract mode. Similarly, recurring tasks such as rollouts may necessitate the utilization of the process mode.

Our main practical implication is that while it would be convenient to define the ways of working in advance—for example, as part of a procurement contract between the parties [68]—and then see the process unfold until the IT artifact is delivered, this is highly unlikely. This means that managers need to be receptive to developments and be able to shift to a different collaboration mode when external conditions or internal issues arise during the course of a project.

Another practical implication is that even in the most well-defined and well-executed projects, the form of cooperation and the communication style must change during the shift from development to maintenance or DevOps [87]. This is particularly emphasized in high-risk projects that are developed in the cooperative or personified mode. In these cases, the collaboration mode needs to shift to the contract mode after the exploration and innovation phase is complete and the development is handed over to a maintenance team.

5.3. Limitations and evaluation

The validity of qualitative research is always difficult to evaluate. Maxwell [88] listed aspects of validity, including descriptive validity, interpretive validity, theoretical validity, generalizability, and evaluative validity. Descriptive validity, or credibility [53], refers to the accuracy of the data—for example, recording the events correctly and accurately reflecting on the events and discussions. We used the same interview protocol for all the interviews but allowed each interviewee to provide detailed descriptions of emerging topics. All the interviews were fully recorded and transcribed, including important nonverbal communication and breaks.

Interpretive validity refers to the researcher's capacity to correctly interpret what the interviewees intended to communicate through their statements and behavior. The first author has followed the development of Birdie since its inception, by working in Integrator in the 1990s. The first author performed the open coding, but the axial coding and selective coding interpretations were iteratively discussed and confirmed with the other authors. In addition, the case descriptions and related interpretations were also discussed with and confirmed by a key stakeholder at Factory.

Theoretical validity refers to the researcher's concepts and the theorized relationships in the context of the phenomena. The essential question is whether the researcher has provided an accurate theoretical explanation of the phenomena. We believe that the identified patterns, concepts, categories, and dimensions fit together well to create a theoretical explanation of the phenomena.

A single descriptive case study cannot be generalized to a population. However, we consider the generalization as theoretical [89], i.e., abstraction of concrete events and actions to theoretical constructs. As for the evaluative validity, where the evaluations made by the researchers are assessed, we suggest extending the study to other cases. Interpretations typically reflect the history and worldview of the researchers.

6. Conclusions

In this paper, we observed the changes in collaboration during a large-scale systems development project, with an industrial corporation as the buyer and a professional services firm as the developer. Together, these parties developed a strategic enterprise system over an extended period of time. Our study found that to succeed in a prolonged collaborative project, the collaboration practices between the parties must shift in response to internal and external incidents, whether they are organizational, technical, personal, or interpersonal. In this case, there were few technical incidents, and these incidents seemed to be easier to manage than the interpersonal and interorganizational incidents. We identified four collaboration modes—the contract mode, the cooperation mode, the personified mode, and the process mode—which differ in their emphasis, requirements, and regularity and can be deployed in reaction to different situations. The study sheds light on how projects can survive and prosper, even after traumatic events and changes in the development environment, by adjusting their collaboration mode accordingly. Collaboration in large enterprise system projects is never static and rigid. Instead, incidents force the project partners to adjust their collaboration mode dynamically to the new situation and its requirements. As most large-scale development is executed through different kinds of outsourcing arrangements, the ability to identify these collaboration modes and guidance on when to apply them is valuable.

The results contribute directly to the practice and theory of enterprise systems development. They emphasize flexible collaboration in enterprise systems development. Collaboration should be considered a dynamic and flexible process, rather than a rigid and preplanned approach as stated in the delivery contract. The results also provide effective tools that practitioners can use to navigate unexpected project incidents and their effects on collaboration. The identified collaboration modes are responses to different kind of crises, incidents, and situations in enterprise systems development. The understanding of how they require and emphasize different elements and are solutions to different problems is valuable knowledge for systems development practitioners.

The theoretical understanding developed here is a step forward in building a comprehensive theory of collaboration in enterprise systems development. As the conceptualization was based on a single case study, more research on different contexts is needed. Studying changes in collaboration entails long-term observation, which will most likely take place retrospectively, as collaboration modes can be difficult to distinguish and define while they are in use. We recommend that the phenomena be approached qualitatively, such as through ethnography or longitudinal archive research. Experimenting with action and design research could provide more insight into this topic. Such studies, no

matter which research method is used, will provide a more in-depth understanding of the complex phenomena of interorganizational relationships and collaboration, particularly within contemporary IS development.

CRediT authorship contribution statement

Kari Smolander: Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Matti Rossi:** Conceptualization, Writing - original draft, Writing - review & editing. **Samuli Pekkola:** Conceptualization, Writing - original draft, Writing - review & editing.

Acknowledgement

Academy of Finland, Finland, grant #259454. Peter Wallenberg Foundation, Finland, grant #PWS2016.0006.

References

[1] Jyt Chang, Etg Wang, Jj Jiang, G. Klein, Controlling ERP consultants: client and provider practices, J. Syst. Softw. 86 (5) (2013) 1453–1461.
[2] S. Sawyer, A market-based perspective on information systems development, Commun. ACM 44 (11) (2001) 97–102.
[3] M. Wiener, M. Mähring, U. Remus, C. Saunders, Control configuration and control enactment in information systems projects: review and expanded theoretical framework, MIS Q. 40 (3) (2016) 741–774.
[4] V. Choudhury, R. Sabherwal, Portfolios of control in outsourced software development projects, Inf. Syst. Res. 14 (2003) 291–314.
[5] Nr Hassan, L. Mathiassen, Distilling a body of knowledge for information systems development, Inf. Syst. J. 28 (2018) 175–226.
[6] Th Davenport, Putting the Enterprise Into The Enterprise System, Harvard Business Review, 1998, pp. 121–131. July/August.
[7] S. Newell, C. Tansley, J. Huang, Social capital and knowledge integration in an ERP project team: the importance of bridging AND bonding, Br. J. Manag. 15 (S1) (2004) S43–S57.
[8] J. Nandhakumar, M. Rossi, J. Talvinen, The dynamics of contextual forces of ERP implementation, J. Strateg. Inf. Syst. 14 (2) (2005) 221–242.
[9] I. Ruuska, T. Ahola, K. Artto, G. Locatelli, M. Mancini, A new governance approach for multi-firm projects: lessons from olkiluoto 3 and flamanville 3 nuclear power plant projects, Int. J. Proj. Manag. 29 (6) (2011) 647–660.
[10] Kj Mayer, Ns Argyres, Learning to contract: evidence from the personal computer industry, Organ. Sci. 15 (4) (2004) 394–410.
[11] A. Gopal, S. Gosain, Research note-the role of organizational controls and boundary spanning in software development outsourcing: implications for project performance, Inf. Syst. Res. 21 (4) (2010) 960–982.
[12] R. Merton, Social Theory and Social Structure, The Free Press, New York, 1968.
[13] P. Hedström, L. Udehn, Analytical sociology and theories of the middle range. The Oxford Handbook of Analytical Sociology, Oxford University Press, 2011.
[14] Rm Davison, Mg Martinsons, Context is king! Considering particularism in research design and reporting, J. Inf. Technol. 31 (3) (2016) 241–249.
[15] A. Momoh, R. Roy, E. Shebab, Challenges in enterprise resource planning implementation: state-of-the-art, Bus. Process. Manag. J. 16 (4) (2010) 537–565.
[16] S. Alshawi, M. Themistocleous, R. Almadani, Integrating diverse ERP systems: a case study, J. Enterp. Inf. Manag. 17 (6) (2004) 454–462.
[17] Y. Yusuf, A. Gunasekaran, Ms Abthorpe, Enterprise information systems project implementation: a case study of ERP in Rolls-royce, Int. J. Prod. Econ. 87 (3) (2004) 251–266.
[18] Tm Somers, Kg Nelson, The impact of strategy and integration mechanisms on enterprise system value: empirical evidence from manufacturing firms, Eur. J. Oper. Res. 146 (2) (2003) 315–338.
[19] Y. Su, C. Yang, A structural equation model for analyzing the impact of ERP on SCM, Expert Syst. Appl. 37 (1) (2010) 456–469.
[20] S. Sawyer, Packaged software: implications of the differences from custom approaches to software development, Eur. J. Inf. Syst. 9 (1) (2000) 47–58.
[21] T. Kähkönen, A. Alanne, K. Smolander, S. Pekkola, Explaining the challenges in ERP development networks with triggers, root causes and consequences, Commun. AIS 40 (1) (2017) 11.
[22] S. Sarker, As Lee, Using a case study to test the role of three key social enablers in ERP implementation, Inf. Manag. 40 (8) (2003) 813–829.
[23] L. Shaul, D. Tauber, Critical success factors in enterprise resource planning systems: review of the last decade, ACM Comput. Surv. 45 (2013) 1–55.
[24] H. Akkermans, K. Van Helden, Vicious and virtuous cycles in ERP implementation: a case study of interrelations between critical success factors, Eur. J. Inf. Syst. 11 (1) (2002) 35–46.
[25] Cp Holland, B. Light, A critical success factors model for ERP implementation, IEEE Softw. 16 (3) (1999) 30–36.
[26] D. Robey, Jw Ross, M.-C. Boudreau, Learning to implement enterprise systems: an exploratory study of the dialectics of change, J. Manag. Inf. Syst. 19 (1) (2002) 17–46.
[27] M. Marabelli, S. Newell, Organizational learning and absorptive capacity in managing ERP implementation projects, in: Proceedings of the International Conference on Information Systems (ICIS 2009), AIS Digital Library, 2009.
[28] C. Brown, I. Vessey, Managing the next wave of enterprise systems: leveraging lessons from ERP, MIS Quarterly Executive 2 (2003) 45–57.
[29] C. Soh, Sk Sia, An institutional perspective on sources of ERP package-organisation misalignments, J. Strateg. Inf. Syst. 13 (4) (2004) 375–397.
[30] Y. Dittrich, S. Vaucouleur, S. Giff, ERP customization as software engineering: knowledge sharing and cooperation, IEEE Softw. 26 (6) (2009) 41–47.
[31] J. Ward, C. Hemingway, E. Daniel, A framework for addressing the organisational issues of enterprise systems implementation, J. Strateg. Inf. Syst. 14 (2) (2005) 97–119.
[32] S. Sarker, S. Sarker, A. Sahaym, N. Björn-Andersen, Exploring value cocreation in relationships between an ERP vendor and its partners: a revelatory case study, MIS Q. 36 (1) (2012) 317–338.
[33] Ls Kirsch, Portfolios of control modes and is project management, Inf. Syst. Res. 8 (3) (1997) 215–239.
[34] Lj Kirsch, V. Sambamurthy, D.-G. Ko, Rl Purvis, Controlling information systems development projects: the view from the client, Manage. Sci. 48 (4) (2002) 484–498.
[35] R. Gulati, P. Puranam, M. Tushman, Meta-organization design: rethinking design in interorganizational and community contexts, Strateg. Manage. J. 33 (6) (2012) 571–586.
[36] G. Juell-Skielse, Cm Lönn, T. Päivärinta, Modes of collaboration and expected benefits of inter-organizational E-government initiatives: a multi-case study, Gov. Inf. Q. 34 (4) (2017) 578–590.
[37] A. Tiwana, M. Keil, Control in internal and outsourced software projects, J. Manag. Inf. Syst. 26 (3) (2009) 9–44.
[38] Rw Gregory, R. Beck, M. Keil, Control balancing in information systems development offshoring projects, MIS Q. 37 (4) (2013) 1211–1232.
[39] N. Levina, Collaborating on multiparty information systems development projects: a collective reflection-in-Action view, Inf. Syst. Res. 16 (2) (2005) 109–130.
[40] J. Damsgaard, J. Karlsbjerg, Seven principles for selecting software packages, Commun. ACM 53 (8) (2010) 63–71.
[41] C. Koch, ERP – a moving target, Int. J. Bus. Inf. Syst. 2 (4) (2007) 426–443.
[42] N. Levina, E. Vaast, The emergence of boundary spanning competence in practice: implications for implementation and use of information systems, MIS Q. 29 (2) (2005) 335–363.
[43] C. Rosenkranz, H. Vranesic, R. Holten, Boundary interactions and motors of change in requirements elicitation: a dynamic perspective on knowledge sharing, J. Assoc. Inf. Syst. 15 (6) (2014) 306–345.
[44] A. Yeow, Sk Sia, C. Soh, C. Chua, Boundary organization practices for collaboration in enterprise integration, Inf. Syst. Res. 29 (2018) 149–168.
[45] O. Volkoff, Ye Chan, E. Peter Newson, Leading the development and implementation of collaborative interorganizational systems, Inf. Manag. 35 (2) (1999) 63–75.
[46] L. Liu, P. Yetton, Sponsorship and IT vendor management of projects, J. Inf. Technol. 25 (1) (2010) 56–64.
[47] M. Tortoriello, R. Reagans, B. Mcevily, Bridging the knowledge gap: the influence of strong ties, network cohesion, and network range on the transfer of knowledge between organizational units, Organ. Sci. 23 (4) (2011) 1024–1039.
[48] M. Skerlavaj, V. Dimovski, Kc Desouza, Patterns and structures of intra-organizational learning networks within a knowledge-intensive organization, J. Inf. Technol. 25 (2) (2010) 189–204.
[49] P. Trkman, Kc Desouza, Knowledge risks in organizational networks: an exploratory framework, J. Strateg. Inf. Syst. 21 (1) (2012) 1–17.
[50] N. Levina, E. Vaast, Innovating or doing as told? Status differences and overlapping boundaries in offshore collaboration, MIS Q. 32 (2) (2008) 307–332.
[51] K. Lyytinen, M. Newman, Explaining information systems change: a punctuated socio-technical change model, Eur. J. Inf. Syst. 17 (6) (2008) 589–613.
[52] Nb Moe, D. Šmite, Gk Hanssen, H. Barney, From offshore outsourcing to insourcing and partnerships: four failed outsourcing attempts, Empir. Softw. Eng. 19 (5) (2014) 1225–1258.
[53] B. Glaser, Al Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research, Aldine, Chicago, 1967.
[54] K. Charmaz, Constructing Grounded Theory, Sage, 2014.
[55] S. Gasson, Employing a grounded theory approach for MIS research. Handbook of Research on Contemporary Theoretical Models in Information Systems, IGI Global, 2009, pp. 34–56.
[56] Cc Pinder, Lf Moore, The resurrection of taxonomy to aid the development of middle range theories of organizational behavior, Adm. Sci. Q. 24 (1) (1979) 99–118.
[57] A. Strauss, J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Techniques, SAGE Publications, Newbury Park, CA, 1990.
[58] Am Pettigrew, Longitudinal field research on change: theory and practice, Organ. Sci. 1 (3) (1990) 267–292.
[59] A. Langley, Strategies for theorizing from process data, Acad. Manag. Rev. 24 (4) (1999) 691–710.
[60] C. Urquhart, H. Lehmann, Md Myers, Putting the 'theory' back into grounded theory: guidelines for grounded theory studies in information systems, Inf. Syst. J. 20 (4) (2010) 357–381.
[61] A. Hanna, J. Windebank, S. Adams, J. Sowerby, S. Rance, A. Cartlidge, ITIL V3 Foundation Handbook, The Stationary Office, Norwich, UK, 2008.
[62] M. Alvesson, D. Kärreman, Constructing mystery: empirical matters in theory development, Acad. Manag. Rev. 32 (4) (2007) 1265–1281.

[63] Fk Alaghehband, S. Rivard, S. Wu, S. Goyette, An assessment of the use of transaction cost theory in information technology outsourcing, J. Strateg. Inf. Syst. 20 (2) (2011) 125–138.
[64] B. Fitzgerald, K.-J. Stol, Continuous software engineering: a roadmap and agenda, J. Syst. Softw. 123 (2017) 176–189.
[65] M. Hoegl, Hg Gemuenden, Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence, Organ. Sci. 12 (4) (2001) 435–449.
[66] Am Aladwani, Change management strategies for successful ERP implementation, Bus. Process. Manag. J. 7 (3) (2001) 266–275.
[67] Km Nelson, Jg Cooprider, The contribution of shared knowledge to IS group performance, MIS Q. 20 (4) (1996) 409–432.
[68] T. Kern, L. Willcocks, Contract, control and 'Presentation' in IT outsourcing: research in thirteen UK organizations, in: Advanced Topics in Global Information Management, 1, IDEA Group Publishing, 2001, p. 227.
[69] E. Carmel, J. Eisenberg, Narratives that software nations tell themselves: an exploration and taxonomy, Commun. Assoc. Inf. Syst. 17 (1) (2006) 39.
[70] Jd Herbsleb, D. Moitra, Global software development, Software, IEEE 18 (2) (2001) 16–20.
[71] Ps Ring, Ah Van De Ven, Developmental processes of cooperative interorganizational relationships, Acad. Manag. Rev. 19 (1) (1994) 90–118.
[72] Sd Pawlowski, D. Robey, Bridging user organizations: knowledge brokering and the work of information technology professionals, Mis Q. 28 (4) (2004) 645–672.
[73] Lj Kirsch, D.-G. Ko, Mh Haney, Investigating the antecedents of team-based clan control: adding social capital as a predictor, Organ. Sci. 21 (2) (2009) 469–489.
[74] R. Sabherwal, The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives, Inf. Organ. 13 (3) (2003) 153–202.
[75] A. Gopal, S. Gosain, Research note—the role of organizational controls and boundary spanning in software development outsourcing: implications for project performance, Inf. Syst. Res. 21 (4) (2009) 960–982.
[76] Pr Carlile, A pragmatic view of knowledge and boundaries: boundary objects in new product development, Organ. Sci. 13 (4) (2002) 442–455.
[77] Tm Somers, K. Nelson, The impact of critical success factors across the stages of enterprise resource planning implementations, in: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, HICSS 2001. IEEE Press, 2001.
[78] H. Tsoukas, R. Chia, On organizational becoming: rethinking organizational change, Organ. Sci. 13 (5) (2002) 567–582.
[79] A. Langley, C. Smallman, H. Tsoukas, Ah Van De Ven, Process studies of change in organization and management: unveiling temporality, activity, and flow, Acad. Manag. J. 56 (1) (2013) 1–13.
[80] A. Hussenot, S. Missonier, Encompassing stability and novelty in organization studies: an events-based approach, Organ. Stud. 37 (4) (2016) 523–546.
[81] T. Hernes, E. Weik, Organization as process: drawing a line between endogenous and exogenous views, Scand. J. Manag. 23 (3) (2007) 251–264.
[82] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (9–10) (2008) 833–859.
[83] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Pearson Education, 2010.
[84] E. Laukkanen, J. Itkonen, C. Lassenius, Problems, causes and solutions when adopting continuous delivery—a systematic literature review, Inf. Softw. Technol. 82 (2017) 55–79.
[85] A. Tiwana, Systems development ambidexterity: explaining the complementary and substitutive roles of formal and informal controls, J. Manag. Inf. Syst. 27 (2) (2010) 87–126.
[86] G. Travis, How the Boeing 737 Max disaster looks to a software developer, IEEE Spectr. (2019).
[87] M. Hüttermann, DevOps for Developers, Apress, 2012.
[88] Ja Maxwell, Understanding and validity in qualitative research, Harv. Educ. Rev. 62 (3) (1992) 279–301.
[89] As Lee, Rl Baskerville, Generalizing generalizability in information systems research, Inf. Syst. Res. 14 (3) (2003) 221–243.

Kari Smolander is professor and Head of Software Engineering Department at LUT University, Finland, and an adjunct professor in Aalto University, Finland. In addition to his long teaching and research experience, he has worked several years in the industry and in 1990s he was the main architect in the development of the MetaEdit CASE tool. His works have been published in journals such as *Information and Software Technology*, *Journal of Systems and Software* and *European Journal of Information Systems*. His current research interests include change in software and systems development practices and software development organizations.

Matti Rossi is a professor of information systems at the Aalto University School of Business. He has been the principal investigator in several major research projects funded by the technological development center of Finland and Academy of Finland. He was the winner of the 2013 Millennium Distinction Award of Technology Academy of Finland for open source and data research. His research papers have appeared in journals such as *MIS Quarterly*, *Journal of AIS*, *Information and Management,* and *Information Systems*. He has been a senior editor of JAIS and Database, and he is a past editor in the chief of Communications of the Association for Information Systems.

Samuli Pekkola, Ph.D., is professor of information systems at Tampere University. He has worked as a visiting associate professor in the Agder University and the Technical University of Delft, and held several positions in the University of Jyväskylä. His research focuses on users in different manifestations of information systems, their management and acquisition, and enterprise architectures. He has been a principal investigator in several major research projects funded by European Union, Business Finland, Academy of Finland, different foundations, or private companies. His research articles have appeared in journals such as *Information Systems Journal*, *Scandinavian Journal of Information Systems*, *Enterprise Information Systems*, *Decision Support Systems*, and *The DATA BASE*. He is currently on the editorial board of four journals in fields of information systems or egovernment, and a past editor in chief of Scandinavian Journal of Information Systems.