Managing the Dynamics of Mutual Adaptation of Technology and Organisation in Information Systems Development Projects

Bendik Bygstad*

The Norwegian School of IT, Schweigaards gt. 14, 0185 Oslo, Norway

This article explores a strategy of mutual adaptation between the information system and the business process, facilitated by the iterative and incremental mechanisms of modern SW engineering frameworks. The concept of mutual adaptation implies that the real innovation of an information systems project is not the software, but the working solution after implementation. Too little is known about this phenomenon: What are the dynamics of mutual adaptation? To what extent is it possible to control? Which organisational mechanisms facilitate it? Should planned mutual adaptation be part of an information systems development project, or is it outside the reasonable scope of such projects?

The article builds on a longitudinal case study from a large public auditing organisation that developed a new audit process in parallel with the construction of a new information system to support it. The project is described through seven iterations as it experiments with different organisational mechanisms to achieve mutual adaptation. The dynamics of mutual adaptation is described by a window of opportunity, where both the information system and the business process are malleable. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: mutual adaptation; information systems development; organisational mechanism

1. INTRODUCTION

The interaction between the information systems development project and the business organisation is a continuous challenge for the software community (Avison and Fitzgerald 2003). This is not only a question of communication and language; on a deeper level, it concerns the sequencing and interaction of change processes in the organisation. When the project is a combination of business innovation and software engineering – how should the two innovation processes be sequenced?

Usually, they are organised in three main steps, as illustrated in Figure 1. First, the business process is analysed and redesigned, and a requirements specification is written to determine the functionality of the software system. Then the software system is designed and constructed, controlled
Figure 1. Traditional sequencing of business innovation and SW engineering

within a software engineering framework. And, finally, the information system is implemented into the business process, along with the training of users. As shown in Figure 1, there is some overlap between the phases (as SW engineering has become more iterative), but the structure remains basically sequential.

This is not always a good strategy. It often leads to misalignments between the organisational context and the information system (Leonard-Barton and Sinha 1993). This happens because it is difficult to specify the requirements at an early stage, and also because both the organisation and the technology may change during the project (Jacobson et al. 1999). The real innovation of an IS project is not the software, but the working solution after implementation (Leonard-Barton 1988). This implies that it is hard to predict the organisational impact of a new information system. It should be developed and implemented in a way that allows for a stepwise learning and adaptation process. Therefore, from a theoretical perspective, a better form is mutual adaptation (Leonard-Barton 1988, Majchrzak et al. 2000), where the business process design and the software engineering are dynamically coupled, and the implementation process is fully integrated (Figure 2).

While theoretically attractive, the mutual adaptation approach is not well understood (Majchrzak et al. 2000) and challenging to achieve in practice (Giaglis 1999). Therefore, there is a need to study this phenomenon over time in order to analyse its temporal and management aspects. This article explores mutual adaptation in modern, iterative information systems development (ISD) projects, addressing the following research questions:

- What are the dynamics of mutual adaptation in ISD projects?
- To what extent is mutual adaptation possible to plan and control?
- Which organisational mechanisms facilitate it?
- Should planned mutual adaptation be part of an ISD project, or is it outside the reasonable scope of such projects?

The rest of the article is structured as follows. In Section 2, the concept of mutual adaptation is defined and discussed. The research method is presented in Section 3. In Section 4, the case is presented. Findings are discussed in Section 5, which also discusses the relationship between requirements engineering and mutual adaptation. Section 6 briefly concludes.

2. THEORETICAL PERSPECTIVES ON MUTUAL ADAPTATION

This section defines mutual adaptation, discusses different perspectives on the question of control and finally describes requirements for an organisational mechanism to control it.

2.1. Mutual Adaptation

Mutual adaptation between technology and organisation is necessary because there will always be misalignments between a technology and a business environment, which must be solved, either by changing the technology or the environment – or preferably both (Leonard-Barton 1988). Mutual adaptation happens both in the development and in the implementation of information systems. Changing the organisation means to change its formal structure or its overall work process or work practices. Changing the technology means to change substantial attributes of the software. Some key features of mutual adaptation are as follows:

2.1.1. Dynamic

The mutual adaptation is a process. Because of the integrated nature of this process, it is a mistake to separate the creation of the product from
the implementation. The mutual adaptation is an ongoing process, fuelled by continuous technological and business change (Leonard-Barton 1988). A change in the organisation may trigger a change in the software, which in turn may trigger a new organisational change. Thus, while a requirements specification represents a one-way process (from the organisation to the technology) and the organisational implementation of technology represent a one-way process in the opposite direction (from technology to the organisation) – mutual adaptation is a two-way process.

2.1.2. Connected to Work Practices
Mutual adaptation is part of the power struggle in any organisation, and is connected more to work practices than to formal structures. In IS-related change projects, a formal approach of introducing a new organisation chart without changing the work practices has often proved unsuccessful. Thus, mutual adaptation requires careful analysis of the congruence between existing and desired work practices (Brynjolfson et al. 1997).

2.1.3. Include the Creation and Transfer of Knowledge
Mutual adaptation includes the creation and transfer of knowledge by establishing strong ties between two different communities of practice (Garrety et al. 2001). In development projects, the IT people need to understand the business issues, and the business people have to understand the capabilities and constraints of the technology. The solution is often negotiated around ‘boundary objects’ (for example, user interfaces), which both communities try to influence.

2.1.4. Emergent, in a Window of Opportunity
Mutual adaptation is emergent, in the sense that it is difficult to plan in detail. The best solution is not an intellectual construction (like a written specification), but a negotiated situation. Thus, a purely design-oriented approach, as presented by (Jacobson et al. 1995), is less likely to succeed. However, timing is crucial because there is usually a relatively short ‘window of opportunity’, where both developers and users are willing to invest the effort to change the work process and the technology (Tyre and Orlikowski 1994). Thus, the mutual learning and adjustment must happen before both sides lose resources and interest.

2.2. Can Mutual Adaptation be Controlled?
Leonard-Barton (1988) found that mutual adaptation should be controlled through successive large and small cycles of alignment. She recommended that developers took some responsibility for the user adaptations, and that the business people should share some of the uncertainty and risk associated to new technology. She also found that the mutual nature of change required that strategic choices should be driven not only from top management but also from the knowledge core of the organisation.

In contrast, the concept of technological drift was introduced to suggest that the mutual adaptation cannot be planned and managed (Ciborra 2000). Technological drift describes a discrepancy between plan and outcome, in respect of the implementation of information technology, in which the organisational implementation of technology is basically unpredictable and unmanageable. Ciborra asserts that the solution cannot be more managerial control, which has proven to be part of the problem, because more control will lead to more side effects.

The software engineering research has not addressed the question directly, but asserts implicitly that mutual adaptation can be planned and managed. The most well-known mechanisms are evolutionary prototyping, and iterative and incremental delivery (Boehm 1988, McConnell 1996). These mechanisms are integrated in current software engineering frameworks like Rational Unified Process (Jacobson et al. 1999), DSDM (Stapleton 2003) and Microsoft Solutions Framework (Microsoft 2001). Although these frameworks are primarily designed to reduce technical risk, they also address the risk of misalignment between the organisation and the software solution. Therefore, iterative and incremental IS projects using these frameworks represent an interesting arena to study this mutual adaptation, because they allow both the developers and the business people to learn, and to act on new learning, during the process.

2.3. Organisational Mechanisms to Manage Mutual Adaptation
If mutual adaptation can be controlled, some kind of management intervention is needed. Several mechanisms have been suggested, like project teams, steering groups, stakeholder webs (Coakes and Elliman 1999), change agents (Markus and Benjamin 1997), relationship managers, and organisational
mechanisms to support user innovations (Nam-
bian et al. 1999). To succeed in achieving mutual
adaptation an organisational mechanism should at
least satisfy three aspects:

- Facilitate learning between different knowledge
  communities: The emergent nature of mutual
  adaptation requires that both groups learn from
  each other (Leonard-Barton 1988).

- Allow the creation of new knowledge: The
dynamic nature of mutual adaptation requires
that both sides develop a thorough under-
standing of each other’s domain areas and con-
straints, to be able to create new solutions (Garrety
et al. 2001, Clarke et al. 2003). New solutions will, in
particular, concern how the use of an informa-
tion system may enable an improved business
process.

- Provide the necessary authority to actually
  implement the necessary changes in both pro-
cesses: If this is not the case, the result may be
good ideas, but no mutual adaptation, which is
dependent on changed work practices (Brynjolf-
sen et al. 1997).

3. RESEARCH APPROACH

The challenge of mutual adaptation was investi-
gated empirically through a longitudinal case study.
The project case was selected from the following
criteria:

1. It should develop customised solutions in large
   organisations.
2. It should use an iterative software engineering
   method.
3. There should be a strong business objective, i.e.
   the software should help to change a business
   process.
4. And, finally, the development team should be
   an experienced one. The reason was that the
   focus of the study was mutual adaptation, not
   on the challenges associated to implementing a
   software process.

Access was granted to software engineering project
aimed at supporting the financial audit process at
the Norwegian Office of the Auditor General.
The research approach was Longitudinal Process
Research (LPR), which aims to study organizational
change over time, through intensive research in
the actual context (Petigrew 1985, Ngwenyama
1998). LPR focuses on building theories strongly
embedded in the context of study. Important criteria
for data collection are (Ngwenyama 1998) as follows:

- Ongoing engagement with the research site to
  observe changes over time.
- Participant observation to contextualise and
  make sense of observations.
- Multiple data sources to record different inter-
  pretations of events and to ensure validity of
  findings.

The case, a software engineering project using
the iterative and incremental software process
Microsoft Solutions Framework, was researched for
16 months, using several techniques for data collec-
tion: An initial workshop with the most important
stakeholders was held to construct a time line for
the project. Project managers, developers, and users
were interviewed at three intervals. Project meet-
ings were observed to understand how problems
were conceptualised and how decisions were made,
and a vast amount of project documentation (plans,
models, reports) was collected. Data was coded
following the guidelines of Miles and Huberman
(1994). After the videotaped interviews were sum-
marised and registered into an Atlas database, texts
were coded with in vivo codes, using only domain
terms. The large volume of project documentation
was coded the same way.

Ngwenyama (1998) suggests three modes for data
analysis: Comprehensive analysis helps identify
underlying structures and patterns of the organiza-
tional process. Temporal analysis helps in contex-
tualising findings by placing events and situations
in a narrative structure. And member verification
ensures that the case description and researcher’s
interpretation are considered correct and meaning-
ful to the organizational actors.

The case was analysed in the following steps:

A time line was constructed, documenting par-
ticipants and technology in each iteration. Then
each iteration of the project was analysed in detail,
while in parallel looking for repeating patterns
and mechanisms. Looking for mutual adaptation,
related terms were recoded, and analysed. Mutual
adaptation was coded to situations, as described
in Section 2, where mutual learning between busi-
ness people and IS people were observed, where
the creation of new knowledge was documented,
or where there was a direct or indirect change of
process structure.
The principle of *member verification* (Ngwenyama 1998) was followed. After the initial workshop with important stakeholders, the timeline with important events and stakeholders was sent to the participants for verification. After a year studying the project, a preliminary case description was written, and commented by stakeholders. And lastly, a validation meeting was held with project managers, developers, auditors and managers, assessing and commenting on the final case description.

4. CASE STUDY: CHANGING THE AUDIT PROCESS AT OAG

4.1. Business Context

The Office of the Auditor General (OAG) is instituted in the Norwegian Constitution. Its main tasks are to audit the central government, ministries and their agencies’ accounts, and to monitor the ministers’ administration of national interests in state-owned companies and banks. The OAG is based in Oslo, where most of the 450 employees are working. The majority of employees are auditors. Auditors perform three types of audit: Financial audit, corporate control, and performance audit. The project described in this article developed a new IT solution for the financial audit, which consists, somewhat simplified, of the following three main steps:

- **Planning the audit**: Receive budget and accounts for the entity. Then classify into account areas, decide materiality limits and assess risks.
- **Execute the audit (visiting the audited entity)**: Produce and follow audit programme, and document the findings.
- **Reporting**: Document conclusions for assessment of entity.

In 1998, a Methodology Group consisting of financial auditors was appointed to improve and standardise the financial audit methodology. Important elements in the methodology were which steps to perform, and defining the areas and scope of the use of individual judgement. New process-oriented standards were implemented in 1999, but it was hard to get acceptance in the organisation. The main reasons for this were assumed to be strong local auditing cultures in the different departments, and lack of IS support.

4.2. The PROSIT Project (‘Process Oriented IT Audit Support’)

A feasibility study in 1998 produced a requirements specification (mainly based on use cases), and assessed five commercial audit support systems. Two of them were tested, but the project found none of them acceptable. The main reason was lack of support for the audit methodology of the OAG, but technical matters were also important. In 1999, it was decided that a tailor-made system be developed. The main objectives for the project were to

- provide a modern and efficient tool to support a standardised audit process;
- support management control of resource use and audit quality;
- support organisational learning through making audit report screens available for the whole organisation.

4.2.1. Software Process and Project Organisation

The IT department had chosen the Microsoft Solutions Framework (MSF) as their process framework, an iterative and incremental framework. (Microsoft 2001). The project organisation was designed to align the organisation and the development, with a strong management and quality focus. An audit department was appointed to be owner of the system, and the other audit departments were represented both in the steering group and the Methodology Group. In March 2000, the project was organised according to the principles of MSF. Six teams were established; customer, user, release, process, development, and QA teams, each of them with a team leader. Following MSF, the team responsibilities and tasks were described in detail, and the teams were empowered to make decisions within their areas.

Prosit was modelled in UML, and programmed in MS Visual Studio in a three-tier architecture. It was run on PC server technology, with a MS SQL database. PC/server communications were based on Citrix terminal server. Prosit was also to be run on a portable PC, using a GSM connection. The solution was developed through five iterations, from March 2000 to July 2002, when the system was set into production. Iterations six and seven were carried out from July 2002 to June 2003. The next section analyses these seven iterations from a mutual adaptation perspective.
4.3. Case Findings and Analysis

Table 1 summarises the case findings, structured according to the MSF iterations. The analysis focuses on the mutual adaptation between the business organisation and the ISD project, and the role (and direction) of the organisational, mediating mechanism. In particular, communication between the two main groups, the creation of new knowledge, and the authority of the mechanism is investigated. The content of Table 1 will be described and discussed in the next section.

4.4. Iterations 1 and 2

The first iteration concentrated on detailing the specification. Several extensive workshops were held, including up to 40 user representatives selected by the departments. The specification needed much detailing, and there were long, open workshops. Coordinating the detailing of use cases, design of screens, and programming and testing of these was difficult with many inexperienced participants. The second iteration developed most of the functionality for audit planning, but the project experienced both cooperation and technical problems.

4.4.1. Facilitating Mutual Learning

The broad workshop approach was chosen to establish a knowledge platform for both the auditors and developers. A member of the Steering Committee commented that ‘The situation was challenging. We did not really know what we wanted, because we had never seen such a system before. On the other hand, the developers did not understand the audit methodology’.

Unfortunately, the approach did not work very well, partly because the complexity of the tasks was too great to handle for the inexperienced participants. Also, some line managers did not prioritise the workshops.

4.4.2. Allowing for Knowledge Creation

As planned, the functionality for audit planning was programmed and assessed. The quality of the GUIs was regarded as being too low, and the specification needed more detailing.

4.4.3. Authority to Influence on the Processes

At this stage, the focus of the project was mainly on technology-related issues. Thus, several decisions concerning the database solution and the design of GUIs were taken. Audit process issues were not really addressed, but it was seen that the audit process description was not detailed enough for the PROSIT design.

At the end of iteration 2, the steering committee was worried. The cost was higher than estimated, productivity and quality lower, and the cooperation between auditors and developers was not tight enough. In a critical evaluation report, a number of measures were specified: The process was given more structure, and the budget was increased. Also, the Methodology Group was drawn heavier into the project.

4.5. Iterations 3 and 4

The objective for the third iteration was to develop a tool and methodology for risk and priority, and to provide the interface to the audit accounts. The objective in the fourth iteration was to produce tools for report findings. A number of successes appeared.

First, the (MSF) roles worked much closer in this iteration. In the first iteration, the user team had defined the requirements, and handed these over to the architect, who then had detailed the architecture for the developers. This created many misunderstandings and delays. Now, the teams worked closer, the auditors often sitting together with developers and prototyping.

Second, the use case descriptions were structured and standardised. From these, the screen designs were solved nicely, establishing a three-part screen structure that set the standard for the rest of the project. As illustrated in Figure 3, it always shows the steps in the auditing process in the left frame of the screen.

This GUI structure allowed for a stronger link between the audit process and the PROSIT design.

4.5.1. Facilitating Mutual Learning

In these iterations, the Methodology Group became heavily involved in the workshops and prototyping. The developers got more detailed feedback on the process, while the auditors learned more about the complexities and constraints of the technology.
Table 1. Adaptation in the PROSIT project

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process</td>
<td>PROSIT approved. Strong management support</td>
<td>Line managers not prioritising project communication</td>
<td>Line managers and auditors more involved in PROSIT</td>
<td>Audit process is changed, as a result of PROSIT design activities</td>
<td>Awareness and training. All departments made their own implementation plan</td>
<td>Courses focusing on the use of PROSIT</td>
<td>PROSIT is widely used as an integrated part of the financial audit process</td>
</tr>
<tr>
<td>Organisational mechanism – and main direction of adaptation</td>
<td>ISD project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UT/PT workshop</td>
<td>UT/PT workshop</td>
<td>MG and PT</td>
<td>MG and PT</td>
<td>PT and IT</td>
<td>PT and IT</td>
<td>PT and line managers</td>
<td></td>
</tr>
<tr>
<td>Requirements spec were detailed</td>
<td>Process unstructured</td>
<td>Schedule not followed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results of iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results of iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results of iteration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process unstructured</td>
<td>Schedule not followed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On target</td>
<td>On target</td>
<td>On time, on target</td>
<td>On time, on target</td>
<td>On time, on target</td>
<td>On time, on target</td>
<td>On time, on target</td>
<td></td>
</tr>
<tr>
<td>Alignment between IS and business process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process methodology implemented in the organisation</td>
<td>PROSIT part of work routine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PT, project team; IT, implementation team; UT, user team; MG, methodology group
4.5.2. Allowing for Knowledge Creation

Conceptually, the group was able to connect the two processes on a practical level: They could translate the audit process into small steps that could be modelled. Detailing the use cases, it was often discovered that the auditing process was not sufficiently described. One auditor commented that ‘The project has improved the methodology in general, because many issues were not really specified. When this was put into screens and procedural steps, these issues had to be decided. The methodology committee then wrote a proposal, which was later decided by the executive group’.

4.5.3. Authority to Influence on the Processes

The initial role of the Methodology Group was to structure the audit process, and it was later drawn into the project to provide information for the requirements specification. This was indeed an ideal situation for mutual adaptation. Members of the group worked in the project group and influenced the project greatly. At the same time, they were also in the position to change the rules of the audit process.

4.6. Iterations 5 to 7

In the autumn 2001, all departments were asked to make their own implementation plan, focusing on which problems were most important to solve. All users were taken through a two-day course held in the departments. In particular, it was focused on audit planning, risk assessment, and accounting requirements. After a two-month pilot period, PROSIT was set into full production in July 2002.

There was some resistance mainly among very experienced auditors who felt the system narrowed their space of individual judgement. A user satisfaction survey in 2002 showed that users thought there were too many screens in the new system. Also, most users spent more time on the audit process than before.

From the viewpoint of the Methodology Group, this was not a real problem – it was a direct consequence of the new audit methodology. The objective of process-oriented audit was to standardise the work process, and, thus, leave less space for individual work styles and judgement. The resistance was not strong, as the key auditors had been enrolled in the iterative process. The Prosit project was also...
very sensitive to user requests and spent a whole iteration (no. 6) to satisfy them.

During spring 2003, the system and organisation stabilised. After the successful iteration 6, focusing on satisfying change requests, iteration 7 focused on management information.

4.6.1. Facilitating Mutual Learning
The extensive training activities in iteration 5 were, in spite of some critics (as noted in the section above), a success for the project, creating awareness and acceptance in the audit departments. The project organisation was reduced in size, but it was maintained during the first year of operation, working on iterations 6 and 7. Providing user support and training, the learning space was kept open because errors and flaws could be corrected quickly.

4.6.2. Allowing for Knowledge Creation
Since each department made and executed their own implementation plan, there was certainly some space for local adjustments, but mainly on minor issues. After implementation, the software change requests had to be assessed by a committee and prioritised. Thus, knowledge creation was on a smaller scale, producing a large number of change requests. These were handled in iterations 6 and 7.

4.6.3. Authority to Influence on the Processes
The implementation team was, naturally, focused on implementing PROSIT into the organisation, and aligning the work process to the system. The main responsibility for implementation was laid on line managers. On the other hand, the ability to influence the development team was now reduced, partly because of budget constraints, and partly because of the bureaucratisation of the change request handling. The window of opportunity for mutual adaptation seemed to be closing.

5. DISCUSSION

This section returns to the research questions. First, the dynamics of mutual adaptation is discussed, and a model is proposed. Then, it is discussed to what extent mutual adaptation is possible to plan and control, and to which degree different organisational mechanisms support it. It is also discussed whether mutual adaptation is within a reasonable scope of SW engineering. Finally, for validity reasons, it is asked if the case could be interpreted alternatively; is mutual adaptation a solution to the problem of poor requirements engineering?

5.1. What are the Dynamics of Mutual Adaptation in ISD Projects?

Mutual adaptation implies that it is possible to change structural properties of both the organisation and the information system. Majchzrak et al. (2000) suggest that it is not the nature of structures (whether it is technological, political or social) that constrains the adaptation process, but rather the malleability of the structure. As shown by Tyre and Orlikowski (1994), this malleability may vary over time, creating ‘windows of opportunity’, where mutual adaptation is possible. The PROSIT case illustrates that the malleability of the information system and the business process varies over time. As illustrated in Figure 4, this may be described in three phases.

As documented in the software engineering research (Jacobson et al. 1999), the malleability of the software product is high at the beginning of a

![Figure 4. The malleability of the information system and the business process](image)
At the beginning of the PROSIT project, it was fairly easy to change requirements, while at the end of the project, it was much harder and more expensive. Thus, the curve in Figure 1 is falling gradually, as more modules of the system are produced. With the iterative and incremental MSF process, the decrease in malleability is not as sharp as in waterfall projects, so there is considerable space for change midway in the project.

The malleability of the business process is generally lower, as it is embedded in institutional practice and culture. During the PROSIT project, the malleability was slowly increasing; the resistance against the specified auditing process was diminishing, mainly as a result of the PROSIT project. After implementation, malleability is gradually decreasing as the changed business process is routinised by the new information system.

In the first phase, the direction of adaptation goes from the business process to the IS in the form of requirements specification: The malleability of the organisation is low, while it is high in the technology. In the last phase, the direction is opposite; the finished information system is implemented into the organisation as a change lever. However, in the second phase, the adaptation happened both ways. This window of opportunity is created because the PROSIT system is in the process of stabilising, while still being malleable. At the same time, the business process is slowly restructuring because of the ISD project.

5.2. To What Extent is Mutual Adaptation Possible to Plan and Control?

A window of opportunity is not sufficient to create mutual adaptation. As argued in Section 2, an organisational mechanism is needed to facilitate mutual learning, allowing for the creation of new knowledge and having the needed authority to implement changes in both processes. The role of the organisational mechanisms in the PROSIT case is summarised in Table 2.

It is important to recognise that participants in a long, socio-technical project like PROSIT are conscious of the emergent nature of such projects, and of the need to establish and observe feedback loops. MSF iterations are the natural frame for doing this assessment. In the PROSIT project, they experimented with different mechanisms, abandoning those that did not work, and reusing those that did. Thus, although user workshops have proven to be successful in many IS projects (Braa and Vidgen 1997, Avison and Fitzgerald 2003), they were not so in the PROSIT iterations 1 and 2. This mechanism was therefore abandoned, and instead the role of the Methodology Group was expanded.

Table 2 shows that all the organisational mechanisms facilitated mutual learning, but only the Methodology Group allowed for the creation of new knowledge and possessed the authority to change both processes. Thus, in a window of time, it was possible to control the mutual adaptation in the PROSIT case.

The extent of this control should not be exaggerated. As has been emphasised in the interpretation of the case, this is not a question of pure design, because the result of the mutual adaptation is basically outside the scope of detailed planning. It is emergent, in the sense that solutions are developed as learning and knowledge creation proceeds in a good project. What is controllable is the organisational mechanism to facilitate this.

What would be the area of validity for this organisational mechanism? Of course, one single case study cannot answer this question, but grounded in Longitudinal Process Research (LPR), it presents some interesting findings. First, as the case shows, it is possible, in a window of opportunity, to facilitate for successful mutual adaptation. Second, the organisational mechanism (represented by the

Table 2. Attributes of organisational mechanisms in the PROSIT project

<table>
<thead>
<tr>
<th>Organisational mechanism</th>
<th>Facilitating mutual learning</th>
<th>Allowing for the creation of new knowledge in both processes</th>
<th>Authority to influence both processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>User workshops</td>
<td>Yes</td>
<td>Yes, but mainly focused on how business requirements influence the IS</td>
<td>No, only the ISD process</td>
</tr>
<tr>
<td>Methodology group</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Implementation team</td>
<td>Yes</td>
<td>Yes, but mainly focused on how IS attributes influence the business process</td>
<td>Partly, but mostly on the business process</td>
</tr>
</tbody>
</table>
Methodology Group) could be copied and used by other businesses.

In line with LPR, it should be emphasised that the findings are context sensitive. In the PROSIT case, the project success was dependent on the Methodology Group, consisting of young and ambitious auditors who realised how the information system could contribute to restructure the business process. They were allowed to spend quite a bit of time in close cooperation with the developers, building a knowledge base. Obviously, this is often not possible. Also, the PROSIT project was run in a well-funded public agency, with moderate time pressures. This allowed the project organisation to experiment with different mechanisms and to extend its project period to one year of operations after implementation. Again, in a more competitive business environment, this might not be an option.

On the other hand, research has long established that close cooperation and knowledge sharing between developers and business people are key success factors (Reich and Benbasat 2000). Even in a more competitive business context, there are equal needs to innovate solutions that include changes in both technology and organisation.

5.3. Should Planned Mutual Adaptation be Part of an ISD Project, or is it Outside the Reasonable Scope of Such Projects?

Should mutual adaptation be part of the ISD project, or be addressed somewhere else (and higher) in the organisation? Admittedly, software development is one of the most complex tasks to undertake for a project manager. Some researchers (for example, Sauer and Willcocks 2004) contend that this challenge is impossible to solve in a project context and have suggested that this should be the responsibility of a new kind of senior manager, an ‘organisational architect’.

Acknowledging that the ISD project manager cannot be solely responsible for mutual adaptation, there are several arguments against this: Organisational change is everybody’s responsibility, and cannot successfully be assigned to a single authority (Markus and Benjamin 1997). Successful mutual adaptation is closely linked to knowledge creation, not only decisions. It is possible to facilitate such knowledge creation (as with the Methodology Group), but its emergent nature makes it hard to plan in detail. Lastly, as the PROSIT case also shows that modern iterative ISD processes (like MSF) have an iterative structure well suited to support mutual adaptation; the iterative and incremental structure extends the period where the information system is malleable.

5.4. Caveat: Can Mutual Adaptation be Replaced by Better Requirements Engineering?

It is reasonable to raise the following objection regarding the PROSIT case: Could the problem of the two first iterations have been solved if the expert Methodology Group had been involved instead of the large user group? Or to put it more crudely: Is mutual adaptation needed only when the requirements process is poorly executed?

Admittedly, there is some truth in this objection because a poor requirements process will increase the need for mutual adaptation. Of course, mutual adaptation does not eliminate the need for SW requirements. However, the objection may be refuted for the following reason.

Requirements engineering is described earlier in the article as one-way adaptation; from the organisation to the information system. It is well known that requirements are unstable, and often impossible to elicit at the start of a project. This aspect was solved by iterative development (Boehm 1988). However, sometimes the problem is that the business process itself is not well designed or implemented. The problem in the PROSIT case was not primarily poor communications between users and developers, but that the business change process did not work properly. In this situation, it was not possible to present good requirements.

Conversely, mutual adaptation is described here as an innovation process; as a process of creating new knowledge both in the business process and the information system. This is quite different than a process of requirements elicitation. In the PROSIT case, this was quite visible as the Methodology Group learned a lot (about the audit process) from the SW development project, which they were able to implement back in the audit process. As observed by Leonard-Barton and Sinha (1993), an initial fit between organisation and technology is not always beneficial; it may imply an automation of outdated routines. Thus, mutual adaptation is not primarily a way to solve a problem (of poor requirements), but an opportunity for innovation.
6. CONCLUSION

This article introduced the concept of mutual adaptation between the SW system and the business process in modern, iterative ISD projects. Four research questions were investigated through a longitudinal case study in a large public auditing agency. The questions and findings are summarised as follows:

- What are the dynamics of mutual adaptation in ISD projects?
  The case study identified a window of opportunity for mutual adaptation. In an ISD project, the malleability of the information system is high at the start of the project; then it gradually becomes lower as more components are produced. Conversely, the business process has a low degree of malleability at the start; then it slowly increases because of the ISD process. At this point of time, the window of opportunity opens, where the malleability of both the information system and the business process is sufficient for mutual change.

- To what extent is mutual adaptation possible to plan and control?
  Mutual adaptation is basically emergent and cannot be planned and controlled in detail. Thus, the interesting question is how the ISD project manager can facilitate and influence mutual adaptation through the use of different organisational mechanisms.

- Which organisational mechanisms facilitate it?
  The case project experimented with three different organisational mechanisms for mutual adaptation: user workshops, an expert process group, and an implementation team. Only the process expert group working closely with the development team was successful in both structuring the information system and restructuring the business process. This was achieved through mutual learning, knowledge creation, and the necessary authority to influence both processes. It is suggested that this organisational mechanism, within the window of opportunity, may be relevant for other ISD projects.

- Should planned mutual adaptation be part of an ISD project, or is it outside the reasonable scope of such projects?
  The ISD project manager cannot be solely responsible for mutual adaptation, but the case illustrates how he or she can facilitate and take advantage of it. Modern iterative ISD processes (like MSF, RUP, DSDM) have an iterative structure well suited to support mutual adaptation; the iterative and incremental structure extends the period where the information system is malleable. The organisational mechanism integrates well with modern software engineering frameworks: The iterative structure extends the period of time where the information system is malleable, and iterations are a natural frame for assessing the mutual adaptation. Therefore, it represents an interesting option for project managers to increase the scope and business value of ISD projects.

ACKNOWLEDGEMENTS

An earlier and shorter version of this article was presented at the Information Systems Development Conference (ISD) in Vilnius, Lithuania, in September 2004.

A special thanks to Liv Dreierstad at the Office of the Auditor General, who provided access to the PROSIT project, and also commented on the case description. I also thank the anonymous reviewers of SPIP for quite useful comments.

REFERENCES


Brynjolfson E, van Alstyne M, Bernstein A, Renshaw AA. 1997. Tools for teaching change management. The matrix of change and supporting software. IAIM 97, Atlanta, USA.


