

Association for Information Systems AIS Electronic Library (AISeL)

ICIS 2008 Proceedings

International Conference on Information Systems
(ICIS)

2008

Dealing With Ambiguous and Fluctuating Requirements of Embedded System Development: A Case-Study

Wolfgang Molnar

University of Warwick, w.a.molnar@warwick.ac.uk

Joe Nandhakumar

University of Warwick, joe.nandhakumar@wbs.ac.uk

Follow this and additional works at: <http://aisel.aisnet.org/icis2008>

Recommended Citation

Molnar, Wolfgang and Nandhakumar, Joe, "Dealing With Ambiguous and Fluctuating Requirements of Embedded System Development: A Case-Study" (2008). *ICIS 2008 Proceedings*. 61.

<http://aisel.aisnet.org/icis2008/61>

This material is brought to you by the International Conference on Information Systems (ICIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ICIS 2008 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

DEALING WITH AMBIGUOUS AND FLUCTUATING REQUIREMENTS OF EMBEDDED SYSTEM DEVELOPMENT: A CASE-STUDY

*Traitement des exigences ambiguës et fluctuantes dans le cadre du développement
de systèmes embarqués : une étude de cas.*

Wolfgang Molnar
University of Warwick
Warwick Business School
Coventry CV4 7AL, UK
W.A.Molnar@warwick.ac.uk

Joe Nandhakumar
University of Warwick
Warwick Business School
Coventry CV4 7AL, UK
Joe.Nandhakumar@wbs.ac.uk

Abstract

This paper presents the findings of a case study that investigates how developers and managers deal with ambiguous and fluctuating requirements during an embedded system development in a structured process management environment. In particular, this paper focuses on improvisation and bricolage actions as a coping strategy by software developers and managers. This research adopts an interpretive approach that involves the collection and analysis of qualitative data. In this study, we observed a turbulent environment with situated improvisational and bricolage responses from developers and managers. The organizational structured process management framework was not sophisticated enough to deal with the existing challenges. Moreover, some improvisational and bricolage activities became institutionalized and, hence, became organizational routines of developers and managers. This paper indicates the value of reflexive practices as vital issues for strategic conduct in the event that improvisational and bricolage activities were deployed as a coping strategy.

Keywords: Information technology requirements, Information system development, Information systems management

Résumé

Ce document présente les résultats d'une étude de cas sur le comportement des développeurs et managers face à l'ambiguïté et la fluctuation des exigences (cahier des charges) dans le cadre du développement de systèmes embarqués. Cette étude se concentre plus particulièrement sur l'improvisation (« bricolage ») en tant que stratégie d'adaptation de la part des développeurs et des managers.

Introduction

The elicitation and management of requirements for software systems is one of the most critical and complex organizational activities within the software development lifecycle (Coulin et al. 2006; Nuseibeh et al. 2000; Reel 1999). Within the software development lifecycle, baseline requirement elicitation, analysis, specification, and validation are the activities typically performed after project initiation and preliminary planning, but before system conception and design (Wiegiers 2003). This initial set of requirements will be updated as new requirements are added or old specifications are revised (McConnell 2006). Hickey and Davis (2003) describe this multifaceted phase of the development lifecycle as ‘learning, uncovering, extracting, surfacing, and/or discovering the needs of customers, users and other potential stakeholders’. Its very multi-disciplinary nature is subject to many factors and a variety of both technical and social issues (Coulin et al. 2006; Jiao et al. 2006; Nuseibeh et al. 2000). When performed poorly, the results usually include intensive rework, schedule overrun, and in some cases, project and system failure (Hickey et al. 2002). Therefore, it is important and necessary to properly conduct the elicitation and management of requirements to avoid additional expense in terms of time, effort, and cost (Coulin et al. 2006; Jiao et al. 2006; Nuseibeh et al. 2000). Some guidelines are provided by the practices for software requirement specifications recommended by the Institute of Electrical and Electronics Engineers (IEEE 1998). This guideline (IEEE 1998) describes the nature of software requirement specifications, whereas the principle concerns are: functionality, external interfaces, performance, attributes, and design constraints imposed on an implementation. Clearly, requirement elicitation and management is highly dependant on the specific project, as well as organizational and environmental characteristics and circumstances (Christel et al. 1992). However, the elicitation and management of requirements are necessary prerequisites for any type of software development (Coulin et al. 2006; Nuseibeh et al. 2000).

Requirement elicitation and management has become even more complex in the context of embedded system development because it bands together the realms of software and hardware. Embedded system development is fundamentally different to nonembedded systems because it addresses specific constraints such as hard timing constraints, limited memory and power use, predefined hardware platform technology, and hardware costs (Graaf et al. 2003). The embedded system developers must understand interdisciplinary product application domain knowledge (Curtis et al. 1988), and physical requirements such as memory, power, and real-time issues (Graaf et al. 2003; Lee 2000; Lee 2002). In addition, many business constraints must be taken into account in industrial new product development environments (Manhart et al. 2004). Gajski and Vahid (1995) claim that an embedded systems functionality is usually fixed and is mainly determined by the system interaction with its environment. However, functionality has steadily shifted from hardware to software (Lee 2000). This paradigm shift from hardware to software has several consequences. For example, it is possible to compile late software specification in response to fluctuating requirements. The use of software also facilitates the reuse of previously-designed functions if the code was written at a processor independent, abstract level (Goossens et al. 1997). Despite these software advantages, traditional software process models (plan/specification driven models, and evolutionary models) need well-defined requirements as well as good management of requirement fluctuations (Kettunen et al. 2005; Sommerville 1996). In addition, structured methods to tackle the elicitation and management of requirements are needed along with several agile methodologies (Fowler et al. 2001), such as Feature-driven development or Extreme programming (Cockburn et al. 2001). Kettunen and Laanti (2005) claim that undisciplined programming (ad hoc, chaotic “hacking”) usually appears differently from disciplined software process models, particular the problem of incomplete, volatile, and uncontrolled requirements. Projects using hacking tend to either skip or obtain a brief set of requirements and, as a result, there is no need to manage the requirements (Kettunen et al. 2005). Furthermore, it is possible to accommodate a certain amount of requirement fluctuations, provided that the key personnel remain in the project (Kettunen et al. 2005). In summary, dealing with the elicitation and management of requirements is a vital part of the software process, and there are no simple shortcuts or magic solutions (Wiegiers 2003).

The worlds of business and technology have become turbulent, high-speed, and uncertain, involving a process of both creating change and responding rapidly to fluctuation (Cockburn et al. 2001). The various methodologies of software process models provide different ways to respond to the challenges of dealing with ambiguous and fluctuating requirements. Although some structured methodologies may inadequately refer to the problem of volatile settings, most plan/specification-driven models and evolutionary models provide some strategies to overcome these challenges (Kettunen et al. 2005). Whereas, agile methodologies emphasize that fluctuation is part of the development process in a turbulent and uncertain environment (Fowler et al. 2001). Improvisational and bricolage activities that were not influenced or less influenced by structured and agile methodologies differ in the way they tackle the challenge with ambiguous and fluctuating requirements. In general, improvisation is positively associated

with accelerated product development in turbulent industries (Akgün et al. 2002). Organizational scholars see improvisation as an effective behavioral strategy for dealing with change, particularly under dynamic conditions (Hmieleski et al. 2006). Improvisational activities are spontaneous, which means there is no time gap between the planning and execution of such activities (Moorman et al. 1998a). Augier and Vendelø (1999) claim improvisation can be considered a rapid problem-solving technique that is used by organizations to cope with surprises. Transcripts of organizational improvisation observations indicated that the individual improvisers did not have a holistic understanding of the entire system. However, the team of improvisers engaged various activities that worked for the system in its entirety. So, the joint improvisations of individual improvisers created a system of improvisational action (Moorman et al. 1998a). Improvisational actions may often involve bricolage, but bricolage also occurs when planning precedes execution (Baker et al. 2006; Cunha 2005; Moorman et al. 1998b). Information system development activities may also involve bricolage (Ciborra 1996), which is defined as making do with the items or resources at hand (Levi-Strauss 1966). Although bricolage may occur in non-improvisational contexts (Miner et al. 2001), improvisation and bricolage are not the same construct. Two main differences exist between the constructs of improvisation and bricolage. First, bricolage may involve gathering and creating something with the items or resources at hand (Levi-Strauss 1966). This physical character of bricolage makes it different from improvisation, which is more general. Second, bricolage may occur with pre-planning, whereas improvisation involves the lack of planning. Additionally, many researchers portray bricolage and improvisation as a coping strategy to secure organizational reliability (Amabile 1988; Cunha 2005). Innes and Booher (1999) describe bricolage as a nonlinear, holistic approach to dealing with difficulty that results in some practical product. Improvisation and bricolage may therefore be seen as applicable practices in turbulent environments (Cunha 2005; Weick 1998).

Our study investigated the problem of dealing with ambiguous and fluctuating requirements during an embedded system development in an ISO 9001 certified organization. While there are many studies on ambiguous and fluctuating requirements in the field of information and system development, this issue has not been explored in detail in an embedded system development context with structured process management. Furthermore, not many studies have investigated the strategic role of improvisation in the context of embedded system development. This paper aims to address these gaps. How do software developers and managers deal with ambiguous and fluctuating requirements in this context and with what consequences? How does the structured process management environment shape the requirement gathering and management practices? What are the implications for the elicitation and management of requirements in such a context? This paper seeks to address these core questions by focusing our research on the improvised and bricolage actions during an embedded system development. We present our theoretical foundation in the next section. The third section will clarify our research methodology and will be followed by a case description and our analysis. This paper concludes with a discussion and implication of our findings for embedded system development.

Theoretical foundation

Although the purpose of this research is to derive a theoretical interpretation from the empirical data (Orlikowski 1993), we also draw on contrasting theoretical pillars to elaborate our theoretical interpretation. The requirement engineering used in sophisticated software process models, such as waterfall, spiral, or agile methodologies, provide various ways in which the process of elicitation and management of requirement can occur in a turbulent and uncertain environment. We pool these software process models together because they imply a certain structure and inhibit a diffuse course of action. On the other hand, improvisational and bricolage activities are in contrast to structured practices and techniques, and provide an ample and comprehensive approach to comply difficulties in an unstable environment.

The three main strands of software process models (specification, evolutionary, and agile development) (Fowler et al. 2001; Sommerville 1996) have a different understanding of the requirement engineering process. The idea of a specification development approach involves piecewise execution of defined steps, which leads to the initial goals of the entire procedure (Ghezzi et al. 1991; Sommerville 1996). For example, the requirement engineering procedure may be characterized by three activities (Gebhard et al. 2000): first, an elicitation phase of the features collects needed data of the requirements (Gebhard et al. 2000); second, during a structuring phase the gathered data is grouped to enable the third and final step of writing a specification of requirements (Gebhard et al. 2000). The goal of an evolutionary development approach is to conduct several cycles of the development process to allow for adjustment in the project timeframe (Boehm 1986). During the cycles, or rounds, the following topics should be clarified (Boehm 1986): objectives, constraints, alternatives, risks, risk resolution, risk resolution results, planning for the next phase and commitment. Therefore, the list of requirements will be compiled in greater detail with every

evolution. Although the agile development approach is a “lighter” approach toward building software (Fowler et al. 2001), it contains several structural guidelines that vary from one methodology characteristic to another. According to the agile manifesto, the agile development approach should provoke an ongoing requirement elicitation process with high involvement of stakeholders (customers, business people, and software developers) (Fowler et al. 2001). This constant gathering and management of requirements is enabled through open communication channels between stakeholders (Fowler et al. 2001). This focus on the stakeholders and communication is the core of agile methods for systems development (Cockburn 2001). The implication is that the creation of system developments is limited by human expression. One procedure in which the system development might work well is called a methodology. This is a set of conventions the development team adapts to suit different situations. The comprehensive frameworks of agile methodologies like XP, Scrum, etc. are tight enough to provide a process and structural curriculum for the education of starters in agile development (Cockburn 2001; Cockburn 2004). We may conclude that the three main strands of software process models have various structural needs in common; for example, stepwise work-off, evolution rounds, or constant interaction with stakeholders.

Improvisational and bricolage activities are in contrast to the structured practices and techniques. These activities are situated and occur during the software process, but are not limited to these development activities. Weick (1998) states that spontaneity and intuition are two vital dimensions of improvisation. Weick (1998) claims that alteration, revision and discovery are purer instances of improvisation because they are an expression of the full improvisational spectrum and because these kinds of changes occur smoothly. In contrast, activities that shift, switch or add are weaker instances of improvisation; this is as a result of unique improvisation and because these kinds of changes occur suddenly. In addition to this classification scheme, Weick (1998) states that previous experiences, current settings, and any given situation provides the pretext for compiling these elements. These pretexts imply tendencies and support some particular development while excluding others (Weick 1998). Mirvis (1998) examines four paradoxes of improvisation: rehearsed spontaneity, anxious confidence, collective individuality, and planned serendipity. These paradox pairs seem to become apparent simply through the creative dynamics of contrary forces. Ciborra (2002) observes three dimensions of bricolage. First, tinkering or fiddling tricks are used to continue development in an unpredictable setting. Second, trial and error activities are applied to extend the amount of experience when actions are necessary in a volatile environment. Finally, bits and pieces are pooled together to accomplish a specific purpose. The pooled items (entities) form a creative collage of resources, which are used in ways for which they were not initially proposed. Ciborra (2002) emphasizes the necessity of all dimensions to gain a competitive advantage. Situated activities are a comprehensive approach to encounter challenges in an unpredictable setting. In summary, improvisation and bricolage are different; but they also share some characteristics such as their occurrence in the continuum between coincidental and methodical activities. Since situated software processes are our focus in this paper, we concentrate on the common features of improvisation and bricolage.

Table 1 lists the three main strands of software process models (specification, evolutionary and agile development) as part of the structured development approach. In addition, the table shows two characteristics of the situated software processes. However, improvisation and bricolage have common characteristics to a certain extent, as bricolage may frequently occur during improvisation (Weick 1998).

Table 1. Development characteristics of structured software process models and attributes of the situated software processes

<p>Table 1. Development characteristics of structured software process models and attributes of the situated software processes</p>

Theme	Differentiation	Author

Structured Software
Process Models

Development
Characteristic

		Evolution	(Boehm 1986; Sommerville 1996)
--	--	-----------	--------------------------------

		Agile	(Fowler et al. 2001)
--	--	-------	----------------------

Situated Software Processes	Attribute	Improvisation	(Ciborra 2002; Mirvis 1998; Weick 1998)

		Bricolage	(Ciborra 2002)
--	--	-----------	----------------

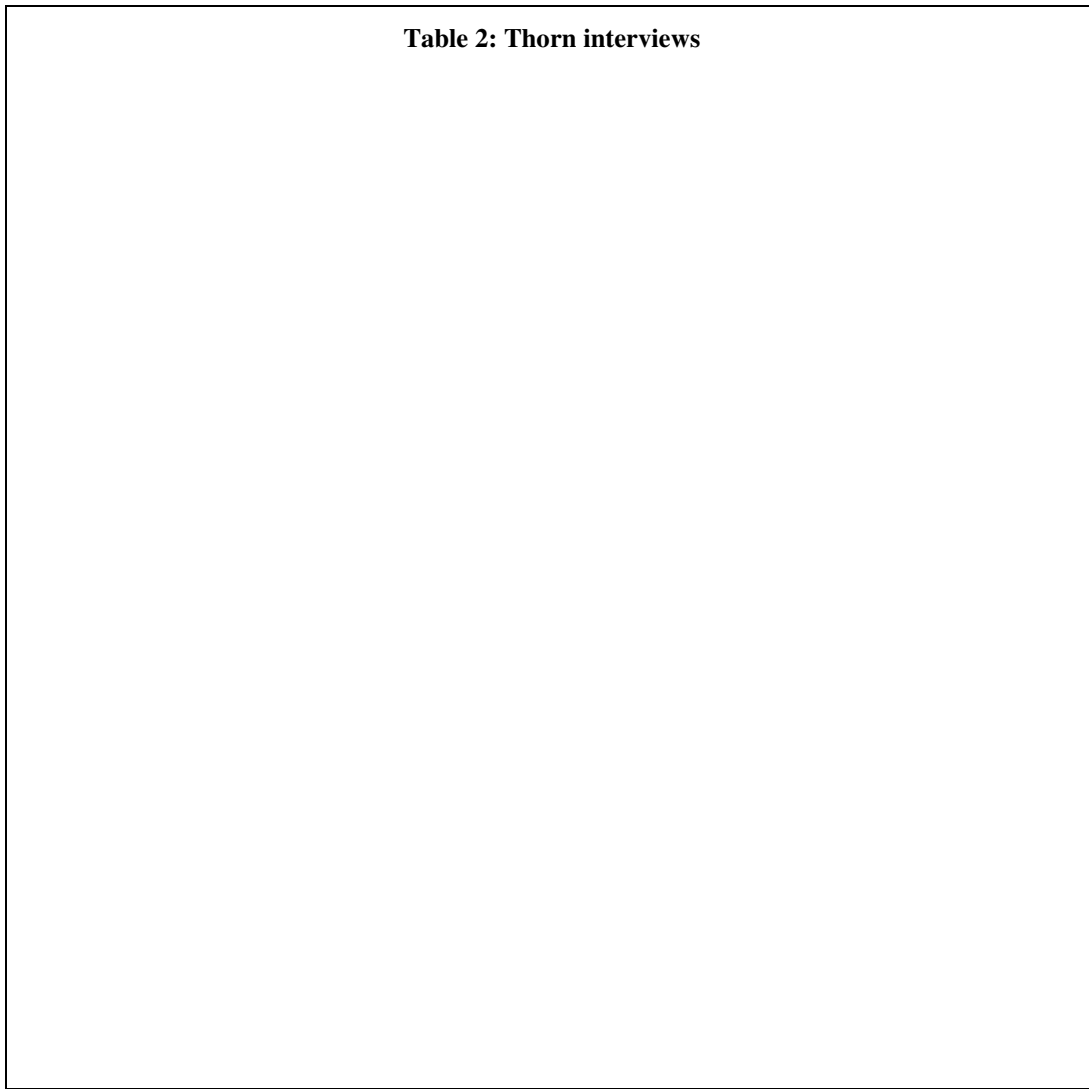
Change is often understood through the altering and revising of work, which may emerge in the improvisation and bricolage of everyday activities (Ciborra 2002; Orlikowski 1995). These continuous activities of alteration and revision may over time become more habitual and routine (Giddens 1986). Human actors often become aware of such habits through conscious reflection or when some form of disruption occurs to routines. Habitual actions create the feeling of security – ontological security – that is convenient and comforting (Giddens 1986). However, human actors may come up with solutions through conscious reflection to overcome problems in adhering to rules in certain contexts. These solutions might cause a break in the routine if the organizational actors cannot follow the rules. These solutions might involve improvisational and bricolage activities. The constituent of the solutions might be articulated thoughts and knowledge in a form that can be shared in an organizational context, hence mainly in

context, hence mainly in discursive consciousness of the organizational actors (Giddens 1986). In this theoretical framework, we try to elaborate our understanding and derive practical implications for the elicitation and management of requirements in a structured process management environment.

Research approach

This research was conducted in an engineering company (P <Thorn> – a pseudonym) in Germany. The researchers were able to collect detailed, qualitative data concerning the context and processes of software developers' actions (Avison et al. 1995; Currie 2004; Walsham 1993). The case study adopted an interpretive epistemological assumption, meaning that the access to reality was through social comprehension (Orlikowski et al. 1991; Yin 1994). Thorn is an ISO 9001 certified organization and develops products for the automotive market. It is large enough to support adequate data for the study and its software developers and managers were motivated to participate in this research. The study was carried out during the development process of an embedded system (ϵ <Epsilon> – a pseudonym). We arranged 19 interviews (two group interviews and 17 semi-structured interviews), each lasting approximately 90 minutes and held in a calm atmosphere (Table 2). In addition, an observer spent an extended period of time with the software development team (description in section "Research Site Context") to make detailed observations of Thorn, its culture, and practices. By performing these informal visits and discussions from January 2007 until June 2007, we discovered the social character of the software developers and their group activities in a natural setting. The researchers strived for a better understanding of the requirement elicitation phase in a structured processes environment. Participants of this research were assured that their statements would remain anonymous and they would not be explicitly linked to their company or product. Most interviews were taped and transcribed, and detailed notes were taken of the observed work practices during each visit.

Table 2: Thorn interviews



1	Semi-structured interview	Product manager	Marketing
---	---------------------------	-----------------	-----------

2	Semi-structured interviews	Middle manager	Software
---	----------------------------	----------------	----------

1	Semi-structured interview	Quality manager	Quality
---	---------------------------	-----------------	---------

1	Semi-structured interview	Senior manager	Development, Finance
---	---------------------------	----------------	----------------------

Total 19	Interviews	p
----------	------------	---

Along with ISO 9001 documents, other records such as meeting reports were qualified and examined as a reference to the observed developer actions. These data sets of qualitative material, which were comprised of observational notes, interview transcripts, and company documentation, were filed electronically. They were read and reread to gain familiarity with the information and to determine trends. Miles and Huberman (1994) refer to the analysis approach applied here as three-tier-coding. The first step, descriptive coding, is exercised to obtain knowledge of the

three-tier-coding. The first step, descriptive coding, is exercised to obtain knowledge of the case's nature by

Case Description

the data. In this way, the researchers are able to bring basic phenomena into the foreground. The second step, called interpretive coding, is conducted to clarify the significance and to relate the context of the

Research Site Context

basic phenomena. The final step - pattern coding - helps to classify data by content of the basic phenomena rather than by the origin of information, and hence to apply patterns. By applying this three-tier-coding procedure to the data, the researchers were able to comprehend the topic of this research on a more in-depth level. The management engaged with the case setting through business field participation. In this research, Thorn's business model consists ofence, pillars to simply establish a bottom-up conceptualization of the obtained and analyzed data, which produces two theoretical abstractions: a significant software process model and a significant software process production. The reputation for creativity and customer orientation. Currently, Thorn employs more than 180 people, and through this enlargement the entire business management underwent major changes, including the establishment of a structured process management framework five years ago. This certificate from an independent organization shows that Thorn implemented a process management framework according to the standards of ISO 9001. Thorn does comply with these standards, because its process management efforts are regularly evaluated by an independent organization. The described standards of ISO 9001 cover processes of the entire organization and are not limited to the product development process. Nevertheless, the product development process was described in great detail with eight multifarious figures, 27 documentation outcomes and 19 links to other documents. This research is largely based on the actions of the middle management, project manager, and associates of the development department.

The director of the hardware and software development was also the senior manager and Thorn's co-founder – we call him Michael. The software development manager was Gabriel (middle management – software, system architecture). Although his main duty was to run the software department (seven software developers at two locations), to a large extent he was also responsible for the system architecture, including hardware and software. As quality manager, David's responsibility is to manage the compliance of ISO 9001 standards through internal audits and support. With the growing success of Epsilon, the tasks of Ester (product manager – marketing) became more and more focused on its marketing. The role of Martin (project leader) was vital for the success of the studied object: the development of a new embedded system. During the initial phase of this development, Martin spent half of his time as a service supplier with the initiator of that new embedded system and half with the developers of this system. As a consequence, he had first-hand contact with the customers as well as the developers. Table 3 lists the various organizational actors whose responses were vital for this study.

Table 3: Organizational actors

--

Name (pseudonym)	Layer	Responsibility

David	Quality manager	Quality
-------	-----------------	---------

Michael	Senior manager	Development, Finance
---------	----------------	----------------------

The development of a new embedded system (ϵ <Epsilon> – a pseudonym) at Thorn was the focus of this

Thorn was the focus of this investigation, and in particular we studied the requirement elicitation and management process management environment. The embedded system was an electronic device based on Linux that used software to connect with the various bus systems of a given test vehicle or test equipment. Its purpose was to do development in the automobile industry. Further, it was possible to connect the system to a laptop or tablet to be used by the customer. However, enabling a relationship between the business field and the development department is a challenge. The influence of Epsilon was a significant change with high expectations of premium cars. To counteract that problem, the availability of a tool to detect these errors by collecting all data was long a dream for many developers in the automobile industry. Martin (project manager) worked at that time to manage the defects concerning the various manufacturers of high-tech devices. As a consequence, he had a profound understanding of the developers' situation. Although some engineering companies were asked to submit proposals for developing such a tool, Thorn got the tender to develop it. Martin was chosen to lead that project, which at the initial phase of product development seemed to be "just another project", as stated by Michael (senior manager). Martin mentioned:

"We started with a relatively simple idea and ended up with a complex product. ... The time during the initial specification was loaded with trouble. We needed to bring together the requests of the customer and the possibilities of reality. After the creation of a successful prototype, they <the customer> tried to implement more features. However, we brought them back to earth. ... At some point we had to decide which features should be developed and which should not. Bringing their desires concerning the Epsilon together was not always easy."

According to Michael, an enormous amount of requirements had to be satisfied during the development of an embedded system within the automobile sector. He informed the researchers that various stakeholders contributed and, as a result, the number of features grew. It is interesting to note that other departments at the customer site joined the original department in ordering Epsilon. These departments have also seen some benefits from Epsilon for their own operations. However, these additional departments had a slightly different focus of operations, which resulted in different requirements for Epsilon. The draft specification had been written shortly after the initial idea. Martin claimed:

"After the first rereads of the draft specification, we needed to improve it. Consequently, the list of requests kept growing."

Several software and hardware developers stated:

"They <the customer> desired the all-in-one device suitable for every purpose. We were lucky that at some point they understood that this was not possible."

Gabriel (middle manager/software) claimed:

"The customer did not really know what he wanted. So he came out with the wise solution to integrate every possible idea. However, we needed to cap these flourishing ideas and put the features into a state which was realizable."

Some of the desired features were truly born in dreamland, because they were hardly or not at all achievable due to technical limitations. Using these different and partly contrasting requirements, it was Martin's responsibility to derive the baseline requirements. Martin's mature understanding of the needs and wants of the test engineers at the customer site was certainly an advantage. Martin stated that he kept asking himself:

"What would I do or have if I was in the situation of a developer in the automobile industry?"

Technical requirements and organizational circumstances

Thorn's hardware development strategy was based on various processor platforms. This strategy was influenced by the rigid environmental requirements (temperature conditions and humidity, etc.) and the usual processor families used in the automobile environment. Although this technical platform was used in a previous Thorn project, some new components were necessary to comply with several issues of the Epsilon requirement baseline. The platform strategy helped Thorn to overcome these difficulties in time. As one developer mentioned, it is interesting that besides the obvious requirements, a lot of referenced requirements (e.g. temperature conditions, user surface perception, etc.) did not make the requirement elicitation and management efforts easier. Furthermore, the customer contributed with varying levels of detail. Martin mentioned that for some parts of Epsilon, the customer was very concerned and specified the requirements in great detail. For example, it was mandatory to use a specific processor

to communicate with the vehicle. The software package of this processor was predetermined and Epsilon developers needed to implement it in the hardware and software. On the other hand, Martin explained that the 32-Bit and the digital signal processor were not regulated at all. In addition, the internal departments at Thorn (marketing, hardware and software department) also had some concerns about the technical requirements. For example, whereas the provided hardware platform initially had a USB adaptor, the software department had no resources to program a USB driver. At some point, Ester (marketing – product manager) decided to leave the USB port out, for economic reasons. Later, the customer inquired about that feature that was unfortunately dropped out of the Epsilon hardware.

Within Thorn, different opinions existed regarding the value of the structured process management framework of ISO 9001. Higher up the hierarchical ladder within the company, the prevalent opinion was that internal motivation was the major reason why Thorn introduced this structured process framework. In contrast, organizational members claimed that external constraints around the company were the reason for introducing the ISO 9001. Developers voiced serious doubts about the practical usefulness of ISO 9001 in their daily tasks. For example, one of the developers claimed,

“Within a good company, ISO is not necessary!”

“Individual motivation and common sense for quality is of greater value than ISO 9001!”

“ISO 9001 raises bureaucracy but is necessary for the company’s products.”

In contrast, other interviewees mentioned the advantages of ISO 9001, such as less friction between the different departments of Thorn. In addition, David (quality manager) claimed that all processes were necessary for the well being of Thorn. Higher in the company hierarchy, the opinion was strong that internal reasons, such as company growth, were the reason for the implementation of ISO 9001. Although the structured processes were established some years before the Epsilon project was started, internal stakeholders still practiced different approaches. However, in many cases, these approaches were applied to the current problem and met the circumstances of that time. Ester explained:

“We did and do a lot of things on top of or besides our standardized processes. A framework for support and service of a complex series product was not established, so we needed to do something about it. We hired people for these responsibilities.”

David stated that some people still needed training and support in complying with the structured process, although the organization is ISO 9001 certified for several years. Gabriel claimed:

“We <developers> at Thorn are a bunch of tinkering enthusiasts who like to fix problems. However, this does not mean that we produce results of poor quality. ... Concerning the requirements, it is worthwhile to mention that Martin put a lot of effort into structuring the various requirements. Nevertheless, the situation during the requirement elicitation phase was quite chaotic and driven by time pressure.”

So, Martin established and fostered a good relationship with the customer. However, a constraining factor during the initial phase of Epsilon was the time pressure, as several interviewees stated. As a consequence, Thorn hired more people to relieve their employees, and the organizational leaders put more effort into interdepartmental procedures to enhance their structured processes, as Michael mentioned. Martin explained:

“I had a list of requirements from the customer and the known platform created by Thorn. I just needed to join them. However, we had some problems in meeting deadlines, because the estimations around that project were way too optimistic. ... I think that the developers should have had a better understanding of the final product. This would have been beneficial for the later integration of the various software parts into one system.”

Managing the complexity

Although some parts of the baseline requirements were not as detailed, the development started. Both Thorn and the customer knew that some issues needed to be redefined during the development. Thorn developers and the project manager reported some contradicting requests from the customer. Therefore, Martin needed to negotiate some of these requests because some of the collected requirements were ambiguous or different to other requirements. A number of features were first mentioned during the development, because at the beginning the customer was unaware of the possibilities, as Gabriel claimed. In addition to this continuous refinement of some requirements, there were also necessary changes. Martin said:

“Well, the changes we performed were within the usual scope of fluctuations. Of course, the developers were unhappy to change or redo something they had already created.”

Many developers claimed:

“The frequent changing of priorities and issues did not make things any easier.”

“Sometimes it was very frustrating to change things quickly to adjust to the latest fluctuation.”

“Frequently we needed to modify some issues.”

With the increasing number of features, the complexity of the latter product grew as well. Furthermore, contrasting wishes of new customers posed another problem when coordinating the development.

In addition to Epsilon’s increasing product complexity, changes inside Thorn were another factor of disturbance. Hiring new people and the introduction of new processes created some distraction at the beginning. Gabriel mentioned:

“We hired some new people during the development. We knew that this would not boost our current development efforts, but we thought more about improving our competencies in the long run.”

Martin stated:

“At the beginning there were only a few people involved. Later, there were more and more people from different departments who had influence.”

Another valuable difference in the Thorn processes was the introduction of a new meeting called change-control-board. This weekly meeting channeled the various requirement fluctuations during the development and maintenance phase. All internal stakeholders were invited to discuss recent modifications of Epsilon’s requirements. These discussions helped the stakeholders to keep up with the recent issues. Gabriel mentioned that the change-control-board enabled new team members to better understand the development processes of Thorn. In particular, Martin claimed that the change-control-board was vital for the success of the Epsilon development.

Analysis

Many themes emerged from the data: ambiguous requirements at the beginning of Epsilon development, fluctuating requirements during the Epsilon development, and diverse influences of structured processes. We think it is important to differentiate between the partly ambiguous requirements that were lacking in various parts, and the fluctuating requirements that caused the adjustment of development activities. The reasons for doing so were the different realms of possibilities of these two situations; nevertheless, both situations comprehended technical challenges. With ambiguous requirements that were lacking in various parts, there was room for interpretation from every side of the stakeholder as these requirements were not specified. However, fluctuating requirements had a describable situation in the past and were adjusted to meet a different setting. Furthermore, the analysis identified diverse influences of structured processes.

Challenging ambiguous requirements

Ambiguous requirements at the beginning of the Epsilon development formed the entire process. These ambiguous requirements were expressed by a requirement baseline of Epsilon, which varied in different detail levels of product features; hence, the comprehended technical challenges. For instance, one processor and its related hardware and software environment were described in detail, whereas the 32-Bit and digital signal processors were not regimented at all. The description of this one processor included fragments of its software core to guarantee interoperability within the tested bus environment. On the other hand, the main processor (32-Bit) and other parts of the hardware and software needed only to comply with some environmental standards, such as temperature and humidity. However, these rudimentary requirements were already implicit in the automotive environment. In addition, the differences between the contributions of several stakeholders were significant because those involved with Epsilon emphasized different product features during the requirement elicitation and management phase. During the initial time of Epsilon, the customer was a harmonized voice with one distinct approach to solving particular problems with the proposed tool. Later in the requirement elicitation and management phase, this harmonized voice became polyphonic because other departments contributed their views on the projected device. In addition to the customer’s polyphony voice, departments inside Thorn had concerns about some features. As explained in the case study, the story about the USB connector was unfortunate in some ways. It is worth mentioning that this emerging landscape

of requirements needed to be elicited and managed by Martin. He did not use any requirement software to manage the gathered requirements; instead he used common sense and his experience as a former service supplier. Therefore, his effort and tacit knowledge during the requirement elicitation and management phase was vital for the market success of Epsilon.

The baseline requirements specified only some rudimentary customer needs, paired with the available choices of an existing technical platform at Thorn. However, the requirements were not specified in detail and some parts needed to be explained at a later stage of development. We found that the developers' usual responses to this challenge of ambiguous or lacking requirements were improvisation and bricolage activities. Despite this situation, the goal of the development was clear for the developers, although only Martin had a sophisticated understanding of that domain. So, Martin was also the contact person for the developers so as to avoid unnecessary activities by the developers. Clearly, parts of Epsilon included a higher amount of improvisation and bricolage than others. This reflects the situation of the baseline requirements, which could have been elaborated upon. Improvisation and bricolage activities were also common to fill the gaps between the knowledge deficits of Martin and the developers. The developers had a better knowledge of technical matters concerning the details of the Epsilon development. As a result, these experienced gaps were covered by improvisational and bricolage activities. These reoccurring improvisation and bricolage activities by the developers also caused a growing complexity because, on top of some improvisational and bricolage, outputs needed to be further improvised and tinkered with. This loop of improvisational and bricolage activities ended when the customer accepted an advanced model of Epsilon. At this time, the customer qualified the technical artifact of Epsilon as the established baseline requirements. However, some minor adjustments were still necessary to comply with the changing environment.

Fluctuating requirements

These fluctuating requirements during the Epsilon development prompted several adjustments of the Epsilon development activities. Even though the baseline requirements were established, the turbulent environment called for several modifications during the development. This turbulent environment was in many cases unpredictable, such as hardware proclamations, slightly different features for the customer, and new Thorn internal requirements for better testing of the embedded system. In the area of hardware development, it is vital to have a stable set of parts to reproduce the same device without any loss of quality. As only one part of Epsilon was running through a proclamation process of its manufacturers, Epsilon developers needed to adjust the technical specification of the first hardware platform to ascertain the availability of all parts of Epsilon. Another fluctuation of requirements resulted in a minor alteration of the customers' requirements. This minor alteration of the previously-accepted advanced prototype of Epsilon brought additional work on parts of the software. Besides this, a request of modification from a different Thorn department caused several adjustments of the requirements for that device and its embedded software. Unfortunately, Thorn focused on practical tests of Epsilon relatively late. These briefly explained examples are just a digest of the numerous fluctuating requirements during the Epsilon development. One of the reasons why Thorn was forced to comply with these fluctuating requirements was the circumstances of the demanding quality standards in the automobile industry as well as the goals they set themselves.

In contrast to the ambiguous requirements at the beginning of the Epsilon development analyzed above, the fluctuating requirements provided the developers with a different technical challenge. The investigated incidents were situated in a relatively well-known circumstance. Although none of the previously-stated fluctuating requirements (hardware proclamation, new customer features, and requirement for testing purposes) were predictable, all of them had a describable prehistory and a clear goal of what should be changed. This condition is in contrast to the ambiguous requirements at the beginning of the Epsilon development. Nevertheless, the given theoretical foundation of situated software processes in the form of improvisation and bricolage matched the analyzed activities of the developers. So, the developers' response to these fluctuating requirements was improvisation and bricolage activities. Such improvisation and bricolage activities were common to comply with the market forces and to refine the software because of fluctuating requirements from various stakeholders. As Gabriel mentioned in one interview, the status of tinkered software was getting more and more *ver-bastelt* (neologism: German prefix *ver-* = forth, pre-, off, wrong and German verb *bastelt* = tinker → further tinkered). As a result, the tinkered software, which was as yet manageable, became more complex. However, Thorn was able to adjust its high-tech product to the new setting with improvisation and bricolage activities in response to fluctuating requirements.

Structured processes

Although the structured processes of Thorn's ISO 9001 framework described some guidelines concerning the requirement elicitation and management phase, the researchers were not able to identify any compliance. Rather, some developers and managers expressed concern regarding its practical help. Martin stated that at the beginning nobody gave any thought to structural guidelines to run the project. Their main concern was to obey the ISO 9001 regulation at the organizational interface for procurement and documented meetings with other organizations. Aside from these concerns, they ignored the standards for their day-to-day work. However, the researchers were able to identify their activities as largely improvisational and bricolage actions. Although it included some risk to rely on one person for the requirement elicitation and management, Martin was capable of rising to the challenges of this task. His independence of software tools for these tasks showed a decent understanding of the necessary actions and practices. The advantage of having this mature understanding of the wants and needs of developers at the first customer site was a critical success factor for Epsilon. In addition, Martin needed to improvise and collect all available information to complete the requirements. However, the fact that a complete list of requirements was never formulated propagated itself in the development lifecycle. This element of improvisation and bricolage was not restricted to the requirement phase, but was observable throughout the entire development. Gabriel put it this way: "The people at Thorn are tinkering enthusiasts, who like to fix problems." The lack of specified requirements concerning the 32-Bit and digital signal processors were overcome with rule-of-thumb estimates by competent contact persons at Thorn. With these estimates and his domain knowledge, Martin was able to create an internal baseline of requirements to begin the Epsilon development. However, the set of requirements remained unstable in some areas and required modification in other parts because the circumstances had changed. These circumstances described the diffuse picture of unknowledgeable or unaware conditions.

Despite the low significance of the ISO 9001 framework for the developers in this context, we were able to observe other forms of structural guidelines with a higher acceptance. The informal work atmosphere and work team culture within Thorn influenced the initial requirement elicitation phase. Much of the information collected from contact persons within Thorn remained unverified and was taken for granted. This trust and confidence in the abilities of Thorn employees was very high during the initial phase of the development phase. The increasing demands of the complex Epsilon development caused some setbacks and, as a result, some new processes were introduced. As described in the case study, the change-control-board was introduced to allow internal stakeholders to discuss the various changes and developments. However, the change-control-board was not only used to discuss the recent fluctuations of requirements, but also to keep up with recent developments for all stakeholders. Moreover, new team members were educated about Thorn's internal processes and the current issues of the complex Epsilon development. So, the change-control-board was a vital part of Thorn's informal structured processes. In addition, new employees were hired to fulfill tasks that had previously been handled inadequately. For example, various stakeholders inside Thorn fulfilled their obligation to communicate effectively with customers. One idea that came out of this new process was the plan to standardize the communication paths to customers. In summary, the structured process framework of ISO 9001 induced the requirement elicitation and management practices so that some reports of meetings with customers were predefined. So, the ISO 9001 framework did not significantly influence the day-to-day activities of the developers or the managers. We found that the organizational ISO 9001 standards provided less practical help for the elicitation and management of requirements. Rather, Martin played a key role as project manager for Epsilon, and although Thorn is a certified ISO 9001 organization, many processes were undefined: the elicitation and management of requirements mostly involved improvisational and bricolage actions. Setbacks and the increasing complexity of Epsilon were the triggers for more structures inside Thorn.

Discussion and implication

The above analysis indicates that improvisation and bricolage actions and practices can contribute significantly to the resolution of master challenges during an embedded system development and particularly at the requirement elicitation and management phase. In addition, we analyzed various process-related factors that influenced the requirement elicitation and management phase. Much of the activities of the developers and managers within Thorn involved improvisation and bricolage. Despite Thorn being an ISO 9001 certified organization, its structured processes did not help the Epsilon development. But, several informal processes emerged from that situation and became established organizational standard processes over time. The emerging processes are highly situated in that particular environment and condition, and foil structured software process models. Although structured software development methodologies pay attention to the problem of volatile requirements, improvisation and bricolage

occur outside the boundaries of these methodologies. Structured software development methodologies do not address all these challenges of volatile requirements, which may occur during an embedded system development project; that means they may not be sophisticated enough to cover these challenges.

Agile methodologies are accustomed to fluctuating requirements during the development lifecycle. However, some proposed procedures are difficult to establish or become unmanageable with growing complexity (Berry 2002; Pinheiro 2003). With the analysis of this case study in mind, we discuss the fact that structured software development methodologies might not be an appropriate answer to blatant problems of volatile requirements. At the beginning of the development, interpersonal relations and communications were important. Previous studies (e.g. (Curtis et al. 1988)) also find that these interpersonal skills are needed for success. However, setbacks and the increasing complexity resulted in changes of the Epsilon development procedures. This restructuring occurred through some situated actions and practices when the developers and managers were aware of needed procedural change. However, we want to highlight the motivation of action (Giddens 1986). The increasing complexity was a severe problem and the developers and managers may come up with rational explanations. Nevertheless, the motivations of action were the unusual circumstances and situations of setbacks, expressed in delayed development deadlines and rework. To cope with the growing complexity, the developers looked for informal and 'tried and tested' organizational standards. The changes applied at Thorn were common instruments to handle turbulent settings (e.g. change-control-board) (Wiegers 2003). The involvement of improvisational and bricolage actions was a routine behavior during the development, although these situated activities might appear as chaotic or lacking in conciseness to outsiders. This routine behavior became experienced and become standards of the organization practices. This routinized behavior improved the self-confidence of their day-to-day activities, and Giddens refers to this as providing Ontological Security (Giddens 1986). So, the developers and managers continued to perform and new institutionalized practices were then drawn on and subsequently became sediment experiences.

To interpret the degrees of improvisation and bricolage as sediment experiences, we trace back to its word origins. Improvisation is paraphrased in Latin as neither quick nor unplanned, but it is called *extemporalis actio* (Ciborra 2002). So, it is action (outside of time), or outside the normal flow of time. From the observed activities, the researchers may deduce the meaning *improvisus* with the meaning "not seen ahead of time" as a more adequate description (Ciborra 2002). Bricolage is paraphrased as making do with the items or resources at hand (Levi-Strauss 1966). The circumstances we describe (ambiguous requirements at the beginning and fluctuating requirements) involve different realms of possibilities. Both situations comprehended technical challenges but the degree of improvisation and bricolage during these situations was another determination factor. Even though the improvisation and bricolage activities with the ambiguous requirements at the beginning of the development involved unforeseen dynamics in a turbulent environment, these dynamics were expected. However, these expected dynamics with the ambiguous requirements involved no preplanning activities according to other structured software process models (specification, evolution, and agile development characteristic). Planned serendipity was mostly comprised during this phase of elicitation and management of requirements. Tinker and trial-and-error activities were applied during this phase to accomplish the goal of a functioning prototype of Epsilon. Most of these activities were self-contained, and motivated further improvisational and bricolage actions. These activities continued until the goal of a functioning prototype of Epsilon was accomplished. In contrast to the ambiguous requirements at the beginning of the development, the fluctuating requirements involved unpredictable dynamics in an unstable environment. The developers and managers of Epsilon became suddenly aware of these fluctuating requirements, each in its different previous history. The fluctuating requirements had in common that a specific goal was defined clearly. Obvious aspects of the improvisational activities during the fluctuating requirements were rehearsed spontaneity and anxious confidence. However, tinker and trial-and-error activities were also practiced during that time. These activities were not self contained, but conditioned by the situation at that time. Its relatedness to a specific goal determined the opportunities, hence the challenges also. The activities were strict and forward-oriented, and Ciborra (2002) calls this passionless actor a problem-solving robot. This differentiation of sudden and not self-contained improvisation and bricolage on one side and expected and self contained improvisation and bricolage activities on the other side, contributes to the discussion on the degrees of improvisation and bricolage, which are vital for a better understanding of the phenomena of situated processes in information system development.

Table 4 describes the situation and insularity of improvisational and bricolage activities. As discussed earlier, the ambiguous requirements situation was expected by the developers and managers at Thorn. As a result of weak specified requirements in that turbulent environment, the improvisational and bricolage activities became self-contained. The activities were therefore not subject of stimuli from outside the group of developers and managers. In addition, Figure 1 depicts the degree of improvisation and bricolage. Situation 'A' symbolizes the improvisational

and bricolage activities of developers and managers with ambiguous requirements during the embedded system development. These activities were self-contained, and they motivated further improvisational and bricolage actions to create a functioning prototype of Epsilon. Situation ‘B’ represents the situated activities brought about by fluctuating requirements. These activities mostly involved improvisation and bricolage and were the response of developers and managers to overcome the challenge of suddenly fluctuating requirements and, hence, to adjust Epsilon.

Table 4: Situation and insularity of improvisation and bricolage			
Example	Situation	Insularity	Quotes of research site members
Ambiguous requirements	Expected	Self-contained	Martin stated: “What would I do or have if I was in the situation of a developer in the automobile industry?”
Fluctuating requirements	Sudden	Conditioned	Gabriel mentioned: “A number of features were clarified during the development. Sometimes, the demanded adjustment was unforeseen and reasoned change or redo of items we had already created.”

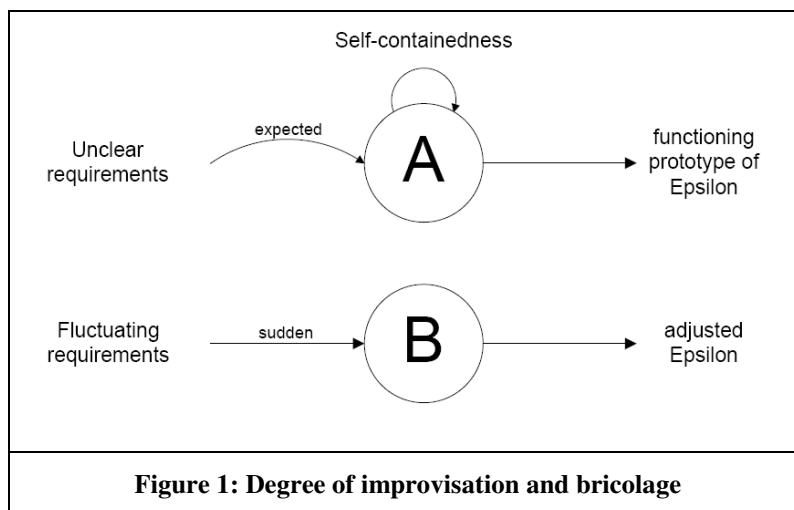


Figure 1: Degree of improvisation and bricolage

In sum, the main finding of this research is that the improvisational and bricolage activities of developers and managers were deployed as a coping strategy to deal with ambiguous and fluctuating requirements of embedded system development. These improvisational and bricolage activities became institutionalized over time, such as the change-control-board or the problem-solving robot (Ciborra 2002). Giddens (1986) claims that this continuity of activities presupposes reflexivity, but reflexivity in turn is possible only because of the continuity of activities that makes them particularly ‘the same’. In this context, reflexivity is the monitored character of the ongoing flow of social life. That basis of improvisational and bricolage activities and reflexivity is in the discursive and practical consciousness of developers and managers. Hence, developers and managers activities were founded on a critical exposure to the respective situation. Although reflexive capabilities of human actors are typically part of day-to-day conduct in the context of social activities (Giddens 1986), we need to emphasize its value for the success of improvisation and bricolage as a coping strategy. To utilize these activities as a strategic conduct, structural scaffolding may provide help to master these challenges. However, the analysis highlights that developers and managers preferred to follow informal standards, whereas the formal structured process management frameworks were ignored mostly by the developers and managers. We believe that guiding and elaborating on the reflexivity of developers and managers might be realized more through informal routines because reflexivity operates mostly in practical consciousness (Giddens 1986). These informal activities occur more on an interpersonal or social level and may stimulate the reflexivity of developers and managers. Orlikowski and Hofman (1997) state enabling conditions

for improvisation in an organization level as aligning key change dimensions and dedicated resources for ongoing support. We however focus more at individual and team levels and suggest that more reflexive practices as a part of the interpersonal strategic conduct of an organization for better use of improvisation and bricolage. For practitioners, this means that the interpersonal as well as the organizational strategic conduct would need to be adjusted if improvisational and bricolage activities are to be deployed as a coping strategy.

Conclusion

This paper investigated the challenge of ambiguous and fluctuating requirements during an embedded system development. From the perspective of the software developers, we tried to understand their comprehension of weak specified and volatile requirements. The research provided insights into the improvisational and bricolage activities of developers and managers. In addition, we answered the question of how the structured process management framework of ISO 9001 induces the activities of developers and managers in that context. This investigation is based on one project in a relatively small company. Future work could address the limitation by investigative case studies about companies of different size and complexity. Furthermore, it would be valuable to investigate how higher compliance with ISO 9001 standards would influence the reflexive capabilities in a similar setting. In this study, improvisational and bricolage actions are examined as a coping strategy by software developers and managers. Some improvisational and bricolage activities became informal standards and over time they became institutionalized. This paper provides better insights into the improvisational and bricolage activities of developers and managers during elicitation, and the management of requirements in a structured process management environment. Moreover, two different degrees of improvisation and bricolage were found, as an expression of its situation and insularity. The paper indicates that reflexivity as a part of the interpersonal strategic conduct of an organization needs to be adjusted and just leveraged if improvisation and bricolage are to be deployed as a coping strategy.

References

- Argüin, A.E., and Lynn, G.S. "New product development team improvisation and speed-to-market: an extended model," *European Journal of Innovation Management* (5:3) 2002, pp 117-129.
- Amabile, T. "A model of creativity and innovation in organizations," *Research in organizational behavior* (10) 1988, pp 123-167.
- Augier, M., and Vendelø, M.T. "Networks, cognition and management of tacit knowledge," *Journal of Knowledge Management* (3:4) 1999, pp 252-261.
- Avison, D., and Nandhakumar, J. "The Discipline of Information Systems: Let Many Flowers Bloom!," IFIP Conference, 1995.
- Baker, T., Miner, A.S., and Eesley, D.T. "Improvising firms: bricolage, account giving and improvisational competencies in the founding process," *Research Policy* (32) 2006, pp 255-276.
- Berry, D. "The inevitable pain of software development, including of extreme programming, caused by requirements volatility.," International workshop on time-constrained requirements, Essen, Germany, 2002.
- Boehm, B.W. "A spiral model of software development and enhancement," *ACM SIGSOFT Software Engineering Notes* (11:4) 1986, pp 14-24.
- Christel, M., and Kang, K. "Issues in requirements elicitation," in: *Carnegie Mellon University Technical Report*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1992.
- Ciborra, C. "The Platform Organization: Recombining Strategies, Structures, and Surprises," *Organization Science* (7:2) 1996, pp 103-118.
- Ciborra, C. *The Labyrinths of information* Oxford University Press, Oxford, 2002.
- Cockburn, A. *Agile Software Development: The Cooperative Game* Addison-Wesley, 2001.
- Cockburn, A. *Crystal Clear. A Human-Powered Methodology for Small Teams* Addison-Wesley Longman, Amsterdam, 2004.
- Cockburn, A., and Highsmith, J. "Agile software development, the people factor," *Software, IEEE* (34:11) 2001, pp 131-133.
- Coulin, C., Zowghi, D., and Sahraoui, A. "A Situational Method Engineering Approach to Requirements Elicitation Workshops in the Software Development Process," *Software Process: Improvement and Practice* (11) 2006, pp 451-464.
- Cunha, M.P. "Bricolage in Organization," in: *FUENL Working Paper Series*, Universidad Nova de Lisboa, 2005.

- Currie, W.L. "The organizing vision of application service provision: a process-oriented analysis," *Information and Organization* (14) 2004, pp 237-267.
- Curtis, B., Krasner, H., and Iscoe, N. "A field study of the software design process for large systems," *Communications of the ACM* (31:11) 1988, pp 1268-1287.
- Fowler, M., and Highsmith, J. "The Agile Manifesto," *Software Development* (9:8) 2001, pp 28-32.
- Gajski, D.D., and Vahid, F. "Specification and design of embedded hardware-software systems," *Design & Test of Computers* (12:1) 1995, pp 53-67.
- Gebhard, B., and Rapp, M. "Requirements Management for Automotive Systems Development," in: *SAE 2000 World Congress*, Society of Automotive Engineers, Inc., Detroit, MI, 2000.
- Ghezzi, C., Jazayeri, M., and Mandrioli, D. *Fundamentals of Software Engineering* Prentice-Hall, Upper Saddle River, N.J., 1991.
- Giddens, A. *The Constitution of Society: Outline of the Theory of Structuration* Polity Press, Cambridge, 1986.
- Goossens, G., Van Praet, J., Lanneer, D., Geurts, W., Kifli, A., Liem, C., and Paulin, P.G. "Embedded software in real-time signal processing systems: design technologies," *Proceedings of the IEEE* (85:3) 1997, pp 436-454.
- Graaf, B., Lormans, M., and Toetenel, H. "Embedded software engineering: state of the practice," *Software, IEEE* (20:6) 2003, pp 61-69.
- Hickey, A., and Davis, A. "The role of requirements elicitation techniques in achieving software quality," REFSQ, Essen, Germany, 2002.
- Hickey, A., and Davis, A. "Requirements elicitation and elicitation technique selection: A model for two knowledge-intensive software development processes," Hawaii International Conference on Systems Science, Big Island, HI, 2003.
- Hmieleski, K.M., and Corbett, A.C. "Proclivity for Improvisation as a Predictor of Entrepreneurial Intentions," *Journal of Small Business Management* (44:1) 2006, pp 45-63.
- IEEE *IEEE Recommended Practice for Software Requirements Specifications* The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1998.
- Innes, J.E., and Booher, D.E. "Consensus Building as Role Playing and Bricolage," *Journal of the American Planning Association* (65:1) 1999, pp 9-26.
- Jiao, J., and Chen, C. "Customer Requirement Management in Product Development: A Review of Research Issues," *Concurrent Engineering* (14:3) 2006, pp 173-185.
- Kettunen, P., and Laanti, M. "How to steer an embedded software project: tactics for selecting the software process model," *Information and Software Technology* (47) 2005, pp 587-608.
- Lee, E. "What's ahead for embedded software," *Computer* (33:9) 2000, pp 18-26.
- Lee, E. "Embedded Software," in: *Advances in Computers*, M. Zerkowitz (ed.), Academic Press, London, 2002.
- Levi-Strauss, C. *The savage mind* University of Chicago Press, Chicago, 1966.
- Manhart, P., and Schneider, K. "Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report," International Conference on Software Engineering, 2004, pp. 378-386.
- McConnell, S. *Software estimation: demystifying the black art* Microsoft Press, Redmond, WA, 2006.
- Miles, M., and Huberman, M. *Qualitative Data Analysis* Sage Publications, Thousand Oaks, 1994.
- Miner, A., Bassoff, P., and Moorman, C. "Organizational improvisation and learning: a field study," *Administrative Science Quarterly* (46:2) 2001, pp 307-337.
- Mirvis, P.H. "Practice Improvisation," *Organization Science* (9:5) 1998, pp 586-592.
- Moorman, C., and Miner, A. "The Convergence of Planning and Execution: Improvisation in New Product Development," *Journal of Marketing* (62:3) 1998a, pp 1-20.
- Moorman, C., and Miner, A.S. "Organizational Improvisation and Organizational Memory," *The Academy of Management Review* (23:4) 1998b, pp 698-723.
- Nuseibeh, B., and Easterbrook, S. "Requirements engineering: a roadmap," International Conference on Software Engineering, Limerick, Ireland, 2000.
- Orlikowski, W.J. "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," *MIS Quarterly* (17) 1993, pp 309-340.
- Orlikowski, W.J. "Improvising Organizational Transformation Over Time: A Situated Change Perspective," in: *Working paper (Sloan School of Management); 3865-95*, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, 1995.
- Orlikowski, W.J., and Baroudi, J.J. "Studying Information Technology in Organizations: Research Approaches and Assumptions," *Information Systems Research* (2:1) 1991, pp 1-28.

- Orlikowski, W.J., and Hofmann, D.J. "An Improvisational Model of Change Management: The Case of Groupware Technologies," *Sloan Management Review* (38:2) 1997, pp 11-21.
- Pinheiro, F. "Requirements honesty," *Requirements Engineering* (8:3) 2003, pp 183-192.
- Reel, J.S. "Critical Success Factors In Software Projects," *IEEE Software* (May/June) 1999, pp 18-23.
- Sommerville, I. "Software process models," *ACM Computing Surveys* (28:1) 1996, pp 269-271.
- Walsham, G. *Interpreting Information Systems in Organizations* Wiley, Chichester, 1993.
- Weick, K. "Improvisation as a Mindset for Organizational Analysis," *Organization Science* (9:5) 1998, pp 543-555.
- Wieggers, K. *Software Requirements* Microsoft Press, Redmond, WA, 2003.
- Yin, R.K. *Case Study Research: Design and Methods* Sage Publications, Thousand Oaks, CA, 1994.