Journal of the Association for Information Systems

Volume 18 | Issue 11 Article 1

11-30-2017

A Paradox of Progressive Saturation: The Changing Nature of Improvisation over Time in a Systems Development Project

Wolfgang Molnar
University of Warwick, wolfgang.molnar@gmail.com

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

Patrick Stacey
Loughborough University, p.stacey@lboro.ac.uk

Recommended Citation
Molnar, Wolfgang; Nandhakumar, Joe; and Stacey, Patrick (2017) "A Paradox of Progressive Saturation: The Changing Nature of Improvisation over Time in a Systems Development Project," *Journal of the Association for Information Systems*, 18(11), .
DOI: 10.17705/1jais.00472
Available at: https://aisel.aisnet.org/jais/vol18/iss11/1

Journal of the Association for Information Systems

Research Paper ISSN: 1536-9323

A Paradox of Progressive Saturation: The Changing Nature of Improvisation over Time in a Systems Development Project

Wolfgang Alfred Molnar
University of Warwick
wolfgang.molnar@gmail.com

Joe Nandhakumar
University of Warwick

Patrick Stacey
Loughborough University

Abstract:

In this paper, we investigate improvisation in a systems development project in the context of safety-critical, rigid quality-management standards. This study took place in a technology company in the automotive industry over a 31-month period and focused on the development of an innovative information system for automobiles. Our analysis traced different forms of improvised practice over the course of a systems development project at the company along with various triggers of improvisation. We found that, as the project progressed, the latitude to improvise became saturated by the increasing structural influences on improvisation. Yet, paradoxically, these structural influences provoked developers to improvise in ways that were progressively more innovative by drawing on accumulated knowledge; we call this phenomenon a "paradox of progressive saturation". We identify ten forms of improvisation that unfold across different stages of a systems development project. We offer a conceptualization of the paradox of progressive saturation to represent the changing nature of improvisation over time, which contributes to the literature on improvisation in information systems development.

Keywords: Socio-technical System, Empirical Research, Interpretive, Improvisation.

Elizabeth Davidson was the accepting senior editor. This paper was submitted August 6, 2014, and went through four revisions.

1 Introduction

This paper reports the findings from a case study of systems development to further our understanding of improvisation in the context of safety-critical, rigid quality-management standards. In this context, system developers face challenges when expected to have a systematic, planned approach to enable control and predictability despite uncertain and emergent requirements. In addition, they experience challenges in having to simultaneously integrate software into emergent hardware innovations, which we see increasingly with Internet of things (IoT) solutions. To respond to such challenges, research has established that improvisation is necessary because planned actions alone are inadequate (e.g., Zheng, Venters, & Cornford, 2011). Research has evidenced this coping potential of improvisation before (Magni, Proserpio, Hoegl, & Provera, 2009) and in safety-critical contexts related to ours, such as emergency and crisis management (e.g., Mendonça, 2007). Prior research in information systems (IS) often treats improvisation as homogeneous throughout a systems development project. For example, researchers have studied improvisation in discrete development stages such as design (Teoh, Wickramsinghe, & Pan, 2012), implementation (Berente & Yoo, 2012), and post-implementation (Rodon, Sese, & Christiaanse, 2011). Yet, we know little about how developers' improvisation practices change across different development stages in the lifetime of one systems development project. As such, we address two research questions:

RQ1: What kinds of improvisation exist in a new systems development project in the context of rigid quality-management standards that are characterized by routinized practices?

RQ2: How do these kinds of improvisation unfold over the course of the project?

We investigated these research questions through a longitudinal, qualitative field study conducted in a technology company in the automotive industry that operated in a safety-critical, ISO-compliant environment. During the field study, we focused on the development practices and improvisations in relation to an innovative, embedded information system. We identified ten major forms of improvised practice at the company. We offer a conceptualization of the changing nature of improvisation over the course of a single systems development project.

This paper proceeds as follows: in Section 2, we review prior research on improvisation in IS. In Section 3, we describe our research design. In Section 4, we present the results of our empirical study. In Section 5, we discuss our contributions to, and the implications for, the existing literature by drawing on the themes that emerged from the findings. Finally, in Section 6, we conclude the paper.

2 Theoretical Foundations

Firms often use quality-management standards such as ISO 9001 in the development of safety-critical software in order to ensure conformity to specific requirements during stages of development such as design, development, production, and installation. Firms establish procedures to control and verify design, which includes conducting planning and design activities, defining organizational and technical interfaces, reviewing and validating design, controlling design changes, and so on. Such process standards are essential for software and systems development that emphasizes rigid quality and safety considerations (Rakitin, 2006; Schrenker, 2006). These standards, broadly based on a document-driven approach to systems development, ensure that the requirements are specified prior to their design and implementation and that documents are only changed through controlled procedures. In this formal setting, improvisation is regarded as an ad hoc, immature, and putatively substandard mode of working (Paulk, Curtis, Chrissis, & Weber, 1993; Bhardwaj et al., 2015). Bodies such as the International Standards Organization (ISO) and the Software Engineering Institute (SEI) often discourage improvisation (e.g., Paulk et al., 1993; Saltz, 2015). They state that, if an organization is improvising, then they are at the lowest level of process maturity: "In an immature software organization, software processes are generally improvised by practitioners and their management during the course of the project" (Paulk et al., 1993, p. 1).

Studies such as Magni et al., (2009) and Zheng et al., (2011) have described plans informed by formal standards as inadequate coping mechanisms as challenges and uncertainties arise that disrupt systems development, and they have evidenced that such circumstances require improvisation. Disruptions take different forms of course; in putatively highly disruptive contexts such as emergency recovery processes, improvisation is reportedly as important as safety, prediction, and planning (Marjanovic & Hallikainen, 2013, pp. 24-32). While this literature has begun to move toward improvisation, it has not necessarily done so away from formality and planning. Rather, a blended view has emerged in which improvisation

and planning co-exist in order to deal with highly disruptive situations. Indeed, in the mainstream IS literature, a common interpretation of improvisation is that it involves overlapping forms of IS work, such as planning, designing, and developing (Effah & Abbeyquaye, 2014, p. 12). The cognate organizational literature on improvisation has expressed this overlapping in terms of dyads, such as conception and execution (Moorman & Miner, 1998a), planning and implementation (Moorman & Miner, 1998b), and real-time planning (Miner, Bassoff, & Moorman, 2001).

A systems methodologist will recognize a similar trait in agile methods, which also feature a variety of activities in small, ongoing iterations (Baskerville, Ramesh, Levine, Pries-Heje, & Slaughter, 2003; Karlström & Runeson, 2005). However, one typically more anticipates these cycles of agile activity despite their fluidity; with improvisation, one typically does not much anticipate what activity comes next (Lanzara, 1999; Louridas, 1999; Weick, 2001; Stacey & Nandhakumar, 2008, 2009). Further, improvisation does not constitute a process model, such as RUP, XP, or Scrum (Abrahamsson, Salo, Ronkainen, & Warsta, 2002), but rather mixes capabilities and in situ actions; that is, one does not model improvisation. The so-called "paradox" literature on IS improvisation has refined this overlapping and blending of IS activities. For example, proposed paradox pairs include "planned serendipity" and "rehearsed spontaneity" (Mirvis, 1998; Zheng, Venters, & Cornford, 2007). These pairs are paradoxical because it seems inconceivable that a developer could simultaneously rehearse and take spontaneous action. This idea is important to our paper, and we extend it by proposing a new paradox pair in our conclusions.

Improvisation is spontaneous in the sense that one performs work in an impulse-driven, "spur-of-the-moment" way. It is based on personal inspiration and autonomous reflexivity in that "actors complete their thinking in relative autonomy" (Mutch, 2010, p. 516). Autonomous reflexivity (Archer, 2007) is characterized by actors' ability to distance themselves from their working environment and, in so doing, not accept that established custom and practice are the "best way" (p. 193); they are selective, evaluative, and elective. They are self-reliant and able to devise courses of action. According to Archer (2007), autonomous reflexivity is particularly applicable when the subject encounters new experiences and novel situations for which their natal context provides no guidelines (p. 194). An autonomous reflexive actor has to learn to rely on their own resources in order to deal with the situation; for example, reflexively generating an innovative solution to overcome a problem created by a breakdown of routine or to exploit an opportunity. For instance, an opportunity arose in an IS under development (Njenga & Brown, 2012) that led to the autonomous, spontaneous reconfiguration of the system with new functionality (McGann & Lyytinen, 2008, p. 4). The spontaneous character of improvisation involves exploring open possibilities where scripted, routine activity does not apply (Bansler & Havn, 2003). Thus, we view improvisation in systems projects as moments in which developers' simultaneously plan and execute actions as they break free or "disembed" themselves from the routine flow of a project with varying degrees of success.

Alternatively, "scripts" may be actively adapted during improvisation as opposed to being ignored outright, which consequently involves less autonomous reflexivity and rather more anchoring in established practice. Research has shown as much during new product developments (NPD) where improvisation involved mixing and matching established procedures in novel ways (Moorman & Miner, 1998b, p. 703; Pavlou & El Sawy, 2010). Indeed, research has shown successful improvisations to draw on accumulated knowledge (knowledgeability) and experience per se (Cunha, Cunha, & Kamoche, 1999; Dybå, 2000; Zheng et al., 2011, p. 6; Weick, 1999; Bansler & Havn, 2003). Improvisation involves developers in an unwitting entanglement with established organizational customs (Archer, 2007) that are part of their repertoire (Zheng et al., 2011, p. 6).

The sway between autonomous reflexivity and established practice yields different levels of improvisation, which generally range from radical improvisational creativity to modest shifts in behavior (Weick, 1993). Moorman and Miner (1998b, p. 703) classify three degrees of improvisation: slight adjustments to preexisting processes, stronger departure from existing practices, and the most extreme level where "the improviser discards clear links to the original referent and composes new patterns". Hence, in its extreme form, improvisation connotes "producing something on the spur of the moment" (Weick, 1998, p. 544), a definition that links back to the idea of spontaneity and autonomous reflexivity. Weick (1998) uses the metaphor of jazz improvisation (cf. Gioia, 1988, p. 66) as a way to illustrate the various "degrees of organizational improvisation". For example, he argues that pure instances of improvisation are activities that alter, revise, create, and discover; that are imaginative and creative; on the other hand, activities that shift, switch, or add are at the less imaginative end of the continuum. Weick (1998, p. 554) claims that jazz improvisation "teaches us that there is life beyond routines, formalization, and success". Activities happen outside organized routines or formal plans (Miner et al., 2001).

In literature review, we also found limited insights into how systems settings characterized by formal quality standards accommodate improvisation. The IS literature often treats improvisation in systems development projects as singular or homogeneous over their lifetime. We extend existing research by studying the nature of improvisation occurrences across multiple development periods in a single safety-critical project. We draw on Archer's (2007) concept of autonomous reflexivity in order to flesh out an aspect of improvisation that research has often pointed to but not unpacked to any significant extent; that is, personal inspiration (Bansler & Havn, 2003; McGann & Lyytinen, 2008, p. 4), which is also an important concept for our case analysis.

3 Research Design and Empirical Context

In this section, we describe: 1) our research approach, 2) site selection and the timeline of the research project, 3) the case study company "ImproCo", 4) the data collected, 5) how we analyzed the data.

3.1 Research Approach

We investigate a systems development process in its context and construct an understanding of the participants' activities in that particular setting over a period of time. As such, we conducted a longitudinal, qualitative field study (Walsham, 1993) at ImproCo (a pseudonym), a high-tech company that operates in Germany's automotive industry. We collected detailed, qualitative data through a combination of interviewing key actors involved in a new technology development project in their natural setting, observing the practices, and reviewing documentation over a 31-month period at ImproCo.

3.2 Site Selection and Timeline

We selected the site after considering "theoretical relevance and purpose" (Orlikowski, 1993, p. 312). We sought to investigate the challenges of the innovative development of new technologies in a "rigid" context of quality-management standards characterized by routinized practices. We selected ImproCo because it was an ISO 9001-certified organization with the relevant contextual elements. For example, senior management implemented the core ISO 9001 processes business acquisition, design and development, test, production, and delivery, and service and support and the supporting processes business management, supplier management, inventory management, and configuration management. The organization's quality manual, which we used to help integrate ImproCo's development processes, documented these processes. ImproCo's senior management regarded adherence to quality-management standards as necessary to help manage organizations through structured processes.

3.3 The Case Study Company

ImproCo provided high-tech systems and services for the research and development centers of automobile manufacturers in Germany. These services and products involved solutions for their customers that ImproCo described as "innovative and tailor-made"; their business success led to an expansion of the company—at the time of writing, ImproCo employed 180 people. Because of the growth of ImproCo, it needed to better align its processes, and ImproCo managers introduced a quality-management standard that was compliant with ISO 9001. ImproCo's quality-management standard covered generic processes for the entire organization and specific ones, including administrative procedures and evaluations of employees and customers. ImproCo's senior management team expected the developers to follow the ISO 9001-certified procedures (clause 7.3 of ISO 9001; i.e., that design and development should have distinct, linear stages). Hence, ImproCo's developers principally followed a linear lifecycle process that proceeded as follows: requirements definition, concept development, initial proof of concept, and, in the final stage, testing and quality checks.

3.3.1 Project Description

Our study at ImproCo focused on exploring the improvisational practices that the organization carried out when developing an in-car information system in a quality-management context. The in-car information system, built on Linux, was designed to help developers collect and analyze data transmissions (i.e., between the various electronic units in a car, such as the central unit and the CD changer) during automobile development. This system involved three different, interconnected subsystems and activities: hardware development, embedded software development, and client software development (data download). These different activities involved various internal stakeholders (software developers,

hardware developers, development managers, project managers, quality managers, and business managers for marketing and customer relations) and external stakeholders (one principal organizational customer with multiple divisions) and necessitated complex discussions and negotiations.

3.4 Data Collection at ImproCo

We interviewed and observed several actors involved in the systems development project that we studied at ImproCo (Table 1). We did so as part of a larger research project that we conducted to better understand improvisation during a new technology development project. One of the actors, Scott (a pseudonym, as with all the names used in this study), occupied a role of critical importance in the project we investigated. The company's employees portrayed him as a "firefighter", someone who helped them whenever they were stuck on a problem. Scott described himself as "a self-taught person with experience and passion". Another actor, Robert, was Scott's supervisor—he was responsible for the development department that comprised 15 people involved in designing hardware and software for this system. Robert engineered the system architecture, which strategically influenced the ongoing development of the in-car IS device. Another actor, Jack, was the project manager; his role was vital to the success of the project because he was also well acquainted with the customer.

Table 1. Data collection at ImproCo

Observations	**Details**			
248 visits	Visits to ImproCo over a period of 31 months			
Number of interviews	**Type of interview**	**Name (pseudonym)**	**Responsibility of interviewees**	**Years of experience**
2	Group	Group	Project under study	
8	Semi-structured	Other developers	Software & hardware development	Ranged between 5 -15 years
2	Semi-structured	Scott	Software & hardware development, "firefighter"	> 15 years
3	Semi-structured	Robert	Development department manager	> 15 years
3	Semi-structured	Jack	Manager of the studied project	> 10 years
2	Semi-structured	Mike	Software developer, graphical user interface specialist	> 5 years
Total: 20				

Figure 1 depicts a timeline of the project and includes our data-collection activities. We conducted 20 interviews (two group interviews and 18 semi-structured interviews) that each lasted approximately 90 minutes at the research site. While ImproCo had more than 180 staff members, we focused on a team of developers and managers that participated in one development project. The first author commenced the field study with two group interviews that involved all developers and managers in order to introduce both himself and the research project. In addition, in accordance with a non-disclosure agreement, the first author assured participants of their anonymity. After these group interviews, the first author conducted 18 semi-structured interviews in which he asked open-ended questions to guide the interview and promote the opportunity to collect rich data through more extensive responses. He carried out the interviews as follows: early in the data–collection process, he conducted interviews that mainly focused on individuals' perspectives of the development process, emerging issues, and the key socio-technical challenges in the project context; towards the end of the data-collection process, he conducted interviews mainly to clarify project participants' perspectives of various observations made (see Figure 1). To support the research process, he tape-recorded the interviews (about 26 hours) and transcribed and translated them from German into English (more than 350 pages of transcription and notes).

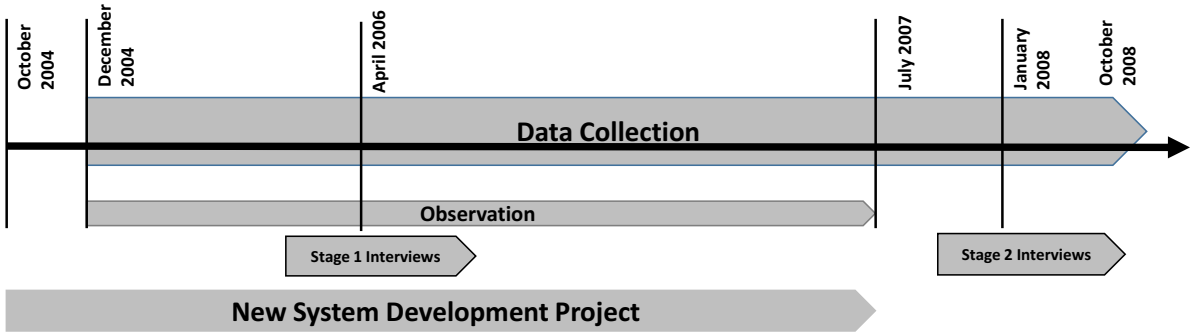

Figure 1. Timeline of the Project with Data-collection Efforts

In addition, between December, 2004, and June, 2007, the first author spent two normal working days a week at the research site with the development team to observe their environment and practices. In total, he visited the site 248 times. Each visit involved mainly observing non-participants (Leidner & Jarvenpaa, 1993; Nandhakumar & Jones, 1997), and he took notes (more than 150 pages in total) of various events and activities of people, including: 1) formal and informal meetings, 2) conversations between developers and/or managers, and 3) behavior during the development activities (mainly related to the project under study). Although he could not observe everything that happened, as the research progressed, he identified and focused on key people and events and uncovered the team's routines in their everyday work setting. As such, he could gain personal experience of the research context under normal conditions and to get behind the official picture (Goffman, 1959). Further, we analyzed company documents to obtain better insights into the context and processes of ImproCo's development activities and to support the review of interview and observation notes.

3.5 Data Analysis

We read and re-read the field notes from observations, interview transcripts, and company documents to become acquainted with the data and distinguish meaningful events and incidents. With the amassed field notes, we could retell the story of ImproCo's development activities in detail (Dyer & Wilkins, 1991).

Broadly speaking, the data analysis process involved three interconnected steps. First, as the data collection progressed, we performed descriptive coding (Miles & Huberman, 1994) of the interview transcripts, observation notes, and other material, such as project and training documentation. In order to address the key research aim, in the initial coding, we focused on identifying and highlighting extracts that described team members' improvisational activities and the changes to their practices throughout the project. We also analyzed the context of these activities and practices, such as associated emergent social and technological constraints that pertained to hardware and software and tensions around ISO procedures and standards.

Second, by examining these codified extracts from interviews and field notes as the analysis progressed, we identified early patterns of instances of different improvisational practices in their context. As such, we identified clusters of improvisational practices (first-order categories) from the data and their related contextual aspects, such as socio-technical issues, over time. In order to understand these emerging patterns in the data, we traced the occurrence during the project of each form of improvisation when analyzing the field data by zooming in (Nicolini, 2009) on developers' day-to-day activities and noting incidents of improvisation and when they ceased to occur.

Third, having identified various forms of improvisation, we further analyzed them in order to develop higher-order categories (second-order themes) that reflected the tensions and "anchors" that related to ongoing practices or developers' spontaneous actions over the course of the project. This process was iterative and nonlinear. Table 2 provides an example of how the analysis process unfolded; Appendix A provides a more detailed version.

Table 2. Stages of Data Analysis

Stage	Analysis	Outputs
1	Initial coding to identify and highlight extracts describing team members' improvisational activities, socio-technical context, and the changes to their practices.	Extracts from interviews and field notes identified as instances of improvisational activities; unfolding of the improvisational practices for the three development periods (Sections 4.1 to 4.3) identified.
2	Identifying early patterns of instances of different improvisational practices in their context and clusters of improvisational forms (first-order categories).	Ten different forms of improvisation; for example, see Table 3 and Figure 3.
3	Identifying higher-order categories (second-order themes) that reflected the tensions and "anchors" that related to ongoing practices or developers' spontaneous actions over the course of the project.	Three levels of improvisation and the unfolding of forms and levels of improvisation over project duration; see Table 4 and Figure 4.

4 Empirical Findings

As we outline above, we focused on the unfolding of the improvisational practices at ImproCo over the course of the development of an in-car information system. We present the process as having three development "periods" (Langley, 1999, p. 703) based on our observation of three major tensions as the process unfolded. Thus, the first period involved following formal plans of action while coping with emerging requirements; the concreteness of the contractual planning activities between ImproCo and its client contrasted with radically changing requirements as the development team carried out prototyping. The second period involved the flexibility to change while stabilizing the system; having established a proof of concept during the first period, the development team more concretely implemented the designs during the second period and, thereby, stabilized the system, yet the client continued to relentlessly request changes to it. The third period involved the developers' finding imaginative workarounds while enforcing quality standards; as the project came to a close with various sign-offs looming, developers' workarounds to problems had to be ever more imaginative. Each of these periods had a cyclical element in terms of ongoing testing, debugging, and review. We depict the timeline and key events per period in Figure 2, which is not meant to be exhaustive but highlights some key events.

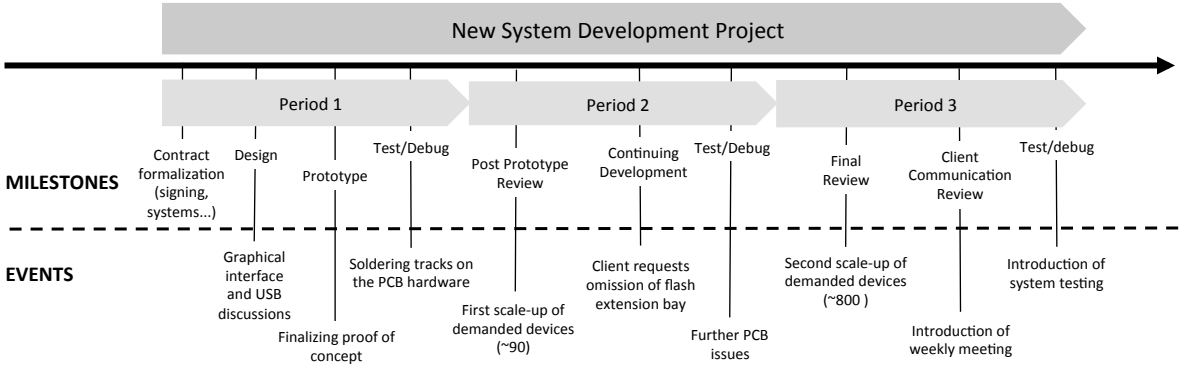

Figure 2. Timeline of the Development Process

While the quality-management context provided strict guidelines and plans for the evolving process, the project involved continuous tensions between following a pre-arranged plan and adjusting to the need of the moment. These tensions constituted a formative context for a variety of triggers and forms of improvisation. We elaborate on these tensions as "roots" of improvisation on a period-by-period basis in the following sections.

4.1 Following Formal Plans of Action while Coping with Emerging Requirements (First Period)

The development process started from a customer's request to ImproCo, which initiated the development of a custom system for that customer. The developers treated each request with great enthusiasm because

they were eager to work on novel and innovative products. Development activities in the automotive industry tend to be highly methodological, so developers mostly follow some sort of routine in their practices that is consistent with quality-management procedures. Some of the formal elements were: 1) formal meetings to discuss budgetary, time, and human resource constraints; 2) client sign-off of agreed milestones, costs to the client, and human resource commitments on the part of the systems developer; 3) implementation and circulation of these agreements to internal management and development teams; 4) assignment of work to developers and the estimation of dependencies; 5) readying of the systems-management environment, such as bug-tracking software to record and manage various system bugs accordingly. The first three formal elements correspond to the core ISO 9001 process of business acquisition, and the fourth and fifth elements correspond to the core ISO 9001 process of design and development. However, not all aspects of development activities were foreseeable, and the developers had to "tinker" with the system. ISO-compliant ImproCo did not officially encourage such tinkering practices, and, as the development process progressed, it became increasingly hard to accommodate them.

The initial period of the project featured many unknowns, and the developers had to respond to fluctuating requirements. They were still defining the proof of concept and "sketching" the embedded system architecture. At this stage, the freshly formed project team found it challenging to make sense of the customer's vague design requirements while following the formal plans and development procedures for the project. For instance, they had no clear requirements for any connectivity ports for the all-in-one device under development. As Robert recalled:

> *The customer wanted to have a USB [Universal Serial Bus] connector at the beginning, and when we stated that this would influence the electromagnetic characteristics of the sensible environment within an automobile, the customer decided to skip this feature. Therefore, we continued further development without a USB…, [but later on] the customer wanted to have this feature again, although the system's architecture provided no availability for a USB connection in terms of hardware or software driver.*

Such fluctuating requirements often forced the project team to change plans, to be more interactive with the customer, and to think ahead. Hence, the developers continuously modified plans while executing them.

Individual developers perceived their key role as providing the best solution for the client even at the cost of breaking some design rules. This self-perceived need to deliver "proper" solutions led to their constantly "tinkering" with and revising the software code before the specifications became stabilized, which helped to address the emerging issues and technical challenges in the customer's context. For example, the developers, of their own choice, revised software code as they became more knowledgeable about the customer's needs and what the customer needed to get the system working. They converted the initial concept into a formal list of requirements for the three different development areas of hardware, embedded software, and client software. Although the developers had extensive experience in developing the hardware and embedded software aspects, they were unfamiliar with developing client software. Robert recalled:

> *I did not expect to extend the client software towards an application with a graphical interface. I thought a line-based console application would just do the job!*

Jack elaborated on the uncertainty:

> *For me, there was no doubt that we would need a graphical interface because the users of this electronic device would like to have a simple solution which does not require additional training. However, the initial ideas did not involve any user-friendly approach, so we needed to adapt to that principal requirement.*

Mike spontaneously developed code for this part of the software while tinkering with the latest models of the physical device that comprised hardware and embedded software. However, the development of these three main system aspects occurred independently and iteratively. Regardless of the original project plan, we observed such voluntary tinkering in the development of the software throughout the project. However, given the strict quality-control regime, the developers did not wish to disclose such contradictory practices to management; they instead gave the impression that software releases for the customer followed the quality procedures. Nevertheless, behind the scenes, a great deal of self-motivation and inspiration triggered different forms of improvisational practice. For example, Scott lost his sense of time during a weekend as he spontaneously and freely examined samples of the hardware while trying to reveal the root cause of the recurrent errors. These errors involved one particular problem that had perplexed the developers for some time: were the errors being "thrown" by the embedded software or the

hardware? Scott claimed that some of the printed circuit board (PCB) tracks were poorly designed and exhibited mismatched processor clock-time and that the "paths" of some integrated circuit pins were too long, all of which caused delays in the signals. In a developers' meeting, they decided to keep that PCB and experiment by soldering some tracks of the hardware. Doing so enabled the developers to continue with their routine work on developing the embedded software. In addition, they decided to find another supplier for the next hardware samples to try to eliminate quality problems. Jack and Mike were unaffected by these problems, however, and they continued to design a new graphical interface. This interface was not part of the initial idea nor was it a customer request, but Jack and Mike saw it as important and developed it from scratch as an innovative solution. While they attempted to develop the interface carefully, they admitted to applying some "shortcuts" for testing purposes. However, they said that these shortcuts did not become part of the official releases. During various coffee breaks, the team discussed Scott's "bold move" in isolating the PCB-related error—they were relieved that the supplier and not they had caused the error. They thought this fact important because they did not want the management to doubt their technical ability to produce high-quality systems. Scott spent the rest of the day in discussions with the hardware developers to further investigate the errors and try to identify a tactic to mitigate them in any new hardware samples provided.

Although ImproCo's ISO 9001-certified procedures and tactics prescriptively shaped their practices, the designers found them less helpful for dealing with the day-to-day design challenges that emerged in the project. The designers sought to appropriate known practices differently to cope with difficulties during the development project, such as frequently changing requirements. For example, technical difficulties arose when the developers created a key feature of the embedded system (although this would have worked fine in a proof-of-concept environment following known practices). Subsequently, they had to twist the known practices to exploit the emerging capabilities. For example, during the project, the entire hardware platform presented an unexpected challenge as ImproCo moved from a 16-bit single-processor platform to a 32-bit multiprocessor platform. The developers had to develop new software code to deal with the technological leap, which created some challenges during the development because the complexity of the hardware, the software, and its interaction all grew. The original agreements did not capture these challenges, meetings had to be held, and budgets had to be flexed as a result.

As the project progressed, other new constraints that resulted from design features based on the incomplete specification emerged. The draft specification was written shortly after the initial idea, and key parts of the requirement were poorly defined and incomplete. Therefore, continuous adjustments of development activities became part of the work during the initial period of the project. Developers increasingly became more pragmatic and tried out new things and often discovered new ways of resolving problems and appended them to the project. Many such discoveries needed to remain somehow in the project scope because of time constraints and the need to realize the project goal. One of the developers commented on this balancing act:

> *We had to follow a moving target. Unclear descriptions of the requirements obstructed our development efforts. Those descriptions were unclear, because the customer did not really know what he wanted. So he came out with the "wise" solution to integrate every possible idea. It was our task to put the features into a state which was realizable. So, we had to investigate and discover features to integrate and features to omit.*

As activities became more structured over time and the design became more stable, the developers found it harder to actualize their new "discoveries".

4.2 Allowing the Flexibility to Change while Stabilizing the System (Second Period)

The next tension we identified in the data relates to seeking the flexibility to change in a stabilizing system. As the project team began to make good progress in resolving much of the ambiguity in the requirements and stabilizing the product specification, they also faced increasing requests for changes from the customer as they moved into the middle period of the development. Hence, during this period, the designers tried to be flexible to alter hardware and software features in response to the customer's ongoing requests and related shifts in circumstances. While trying to stabilize the data storage aspects of the system, the customer's ambivalence necessitated the developers' flexibility as Jack explained:

> *Another challenge was the undecidedness of the customer, which resulted in the problem of a moving target. For example, besides the issue with the USB connector, we initially planned to*

> *include a Compact Flash connector for the device, so that users might have been able to extend the data storage capability by flash memory. Although the project stakeholders agreed on the reasonableness of the extension bay, we needed to skip the Compact Flash connector at an advanced stage of the development because the customer suddenly decided to leave it out.*

A fortunate consequence of the episode that Jack described was that it stabilized the hardware architecture. However, the developers still need to tailor the software architecture to the exact technical setting of the system's various chips and processors. In this context, the tailoring of the embedded software architecture required frequent tinkering and learning by doing until the developers (and their managers) were sufficiently confident of their solution.

In order to handle the increasing numbers of customer requests and as the products became more complex and challenging, the managers and developers decided to change design practices in an effort to adapt to the new situation. For example, Jack recalled:

> *This project was not planned to develop a system for mass production. The prototype development involved only a handful of devices. After we had gotten the supplier's contract, we talked about 80 or 90 devices. However, we are now talking about 800 devices and we originally thought that 80 or 90 devices would be the maximum. So, that's a huge difference, not only in terms of development, marketing, and production, but also in terms of maintenance. Indeed, over time we were able to adapt our development approach to this increasing number.*

The team had to scale up and adapt their practices as they dealt with the emerging situation. Although they had a limited ability to prepare and plan for all eventualities, managers and developers were able to channel their knowledge in response to the emerging challenges. For instance, the hardware element was becoming more stable and they were aware of the potential side effects of increasing the system's complexity. As such, they had to respond quickly to new demands and challenges in order to keep the project on schedule. A good example is how the graphical interface team dealt with incoming customer requests that disrupted their workflow. Customers asked for advice on such things as working with the software, incidents with the software, and solutions to bugs. Fortunately, the project team maintained a bug-tracking system so that they recorded and could manage arising complexities such as incidents, requests, and bugs accordingly. We describe another example of responding quickly to new demands and challenges surrounded the PCB problem above. Continuous interruptions to deal with necessary administrative work and discussions with hardware suppliers, other developers, and the project manager all thwarted Scott's attempts to resolve it. Although he managed to devise an innovative workaround, he explained to the team that it was only a temporary solution, which could potentially introduce further problems.

4.3 Finding Imaginative Workarounds while Enforcing Quality Standards (Third Period)

The next tension we identified in the data relates to imaginative and innovative workarounds in an ISO-oriented culture. As the new system neared completion and with various sign-offs looming, the customer became even more engaged and was making several last-minute requests for change. Interactions with the customer became very tense as existing practices for responding to such requests became disruptive to the work context. Therefore, the team rapidly devised imaginative ways of responding to urgent requests for system changes. Jack explained:

> *On a whim, I invited the representative stakeholders to discuss urgent project issues. The growing complexity required this meeting in order to find best approaches for further progressing [the project]. In addition, we required another approach to stay on schedule. I realized that the way we communicated during the project was getting unmanageable. In order to get things done more properly, I started to invite the project stakeholders for meetings, so that we were able to discuss the immediate issues and decide on how to proceed. Those issues involved just another change in the requirements and the discussion of additional system features. However, not all issues were resolved and, over time, it became a standard procedure in the form of a weekly meeting, which now is part of our work setting.*

This situation was imaginative in the following sense: at a point in the process when the project became ever more structured and with ISO-based sign-offs looming, one would not expect such a radical meeting to take place. Yet, Jack realized the need for the meeting "on a whim" as he put it. The tense interaction with stakeholders also inspired the team to experiment with different ways to organize activities such as

customer support. For example, Jack tried out different modes of interaction with customers and rearranged responsibilities to make this process better. This inspired new ways of handling interactions with stakeholders and freed up developers' time to deal with the design work more efficiently. However, the growing stabilization of the system increasingly constrained their latitude to find effective workarounds. For example, until this point in the project, the individual developers had tested their own code for the various system components for which they were responsible, such as the controller area network (CAN) chip, digital signal processing (DSP) chip, embedded Linux software, and client software. This approach to testing had been successful in previous projects for the company, so it seemed normal to them, but the current project was becoming increasingly complex. In order to coordinate testing practices and raise the quality of the product, the senior developers decided to make team testing more regimented by introducing system-integration testing.

As we outline above, despite all their efforts to resolve matters "on the hoof", they were not always successful; malfunctions sometimes occurred for no apparent reason, although the increasing complexity of the system may have played a part. As a result, individual "star developers" had to come to the rescue by creating solutions spontaneously.

Developers and managers considered Scott an important team member because he was able to deal with project challenges spontaneously. Although most developers had their "eureka" moments, the developers and managers were particularly impressed by Scott's effort in spontaneously solving hardware and software problems throughout the project but particularly during this critical, latter stage of the project when resource budgets were running out. Scott admitted, however, that his skills were not confined to know-how from the hardware and embedded software domains but also reflected an intense desire and dedication to finding the cause of a problem. He described his approach not as "tinkering" but as "goal-oriented tasking; somewhere between creative chaos and structure".

4.4 Summary

The findings above that pertain to three periods of the development project highlight the key forms of improvisation in a context that comprised a variety of socio-technical challenges, such as rigid quality-management standards, ongoing shifts in customer circumstances, routinized practices, and hardware and software constraints. Table 3 summarizes these findings in terms of the associated forms of improvisation and their triggers, contexts, and anchors. Table 3 shows the 10 forms of improvisation that we could induce and that arose from tensions in the development process and context; the table also shows case examples of triggers for each form of improvisation (column 2). In the fourth column, we introduce "anchors of improvisation"—conceptual descriptors for each improvisational form; drawing on concepts from the literature review, they reflect whether each respective improvisational form was based on autonomous reflexivity, ongoing practices, or a mixture of both. We discuss this table further in the next section.

Table 3. Triggers of Improvisation at ImproCo

# Form of improvisation	Case example of triggers	Socio-technical context	Anchors of improvisation
Tension: Following plans of action/Emerging requirements (first period)			
1. Planning while executing	Scheduled work interrupted by fluctuating customer requests during the USB episode.	Fluctuating requirements creating a "second guess" environment of features customers might need (e.g., USB port that interrupted scheduled work for a time).	Mix of autonomous reflexivity and ongoing practices.
2. Revising voluntarily	Developers' self-perceived need to deliver proper solutions for the customer (e.g., the anticipation that the customer would prefer a graphical interface for the software client).	Developers became more knowledgeable of customer needs while the hardware and software were still malleable.	Based on personal inspiration and autonomous reflexivity.
3. Appropriating known practices differently (remixing)	ISO 9001 tactics were less helpful and the developers had to appropriate practices differently, using common sense, as the complexity of the hardware/software grew.	Strict ISO regime and challenging 32-bit hardware platform.	Based on ongoing practices.

Table 3. Triggers of Improvisation at ImproCo

4. Discovering and appending	Unknown/unclear initial requirements—developers decided to integrate every possible idea to discover features, then decide whether to integrate or omit them.	Pressure to realize project goal while the hardware and software still afforded development of new features.	Based more on personal inspiration and autonomous reflexivity.
Tension: Allowing flexibility to change/Stabilizing system (second period)			
5. Altering design features in response to shifting circumstances	Although the customer originally wanted the feature that extends the data storage capability by flash memory, they later decided they did not want it.	Indecisive customer and hardware challenges.	Based on a mix of autonomous reflexivity and ongoing practices.
6. Changing design practices and adapting to new situations	The volume of the devices demanded by the customer was massively revised—the team had to scale up their design practices to cope.	Shifting customer needs/development circumstances and complexity of the hardware and software.	Based on a mix of autonomous reflexivity and ongoing practices.
7. Channeling knowledge to respond to emerging challenges	Refined requirements and understanding but growing complexity.	Acquiring practical insights about the contextual issues of customer environment and better sense-making of technical aspects.	Based more on ongoing practices.
Tension: finding imaginative workarounds/enforcing quality standards (third period)			
8. Devising responses to urgent needs	Communication overload with customers due to their increased demands. Weekly meeting routine established.	Tense interaction with stakeholders while the system was becoming more stable.	Based more on ongoing practices.
9. Experimenting with work organization	Having set up a weekly meeting routine with customers, experimenting with work organization inspired a new approach to coping with customer support, too. Concurrently, system integration testing also introduced a more structured approach.	Tense interactions and structured work organization/expectation.	Based more on ongoing practices.
10. Creating solutions spontaneously	Responding immediately to malfunctions (e.g., the "paths" of some connected pins from different integrated circuits were too long, which meant the signal transmission was delayed and caused malfunctions).	Management style supporting individual problem solving and approach to time and novel technology experimentation.	Based on personal inspiration and autonomous reflexivity.

As Table 3 summarizes, each of the 10 forms of improvisational response was anchored in autonomous reflexivity, ongoing practices, or a combination of the two. These forms of improvisation were also broadly associated with specific periods of the project. By combining the anchors and periods in Figure 3, we depict the unfolding patterns of the improvisational forms at ImproCo. One should consider Figure 3 more as a "floor sketch" of movements over the project's duration rather than as a graph. Hence, the downward slope of the lines indicates improvisational forms' changing their anchors over time from autonomous reflexivity to becoming more related to ongoing practices. The improvisational forms at the top of the figure are more anchored in developers' autonomous reflexivity. We discuss Figure 3 in this section according to the three periods and three associated anchors.

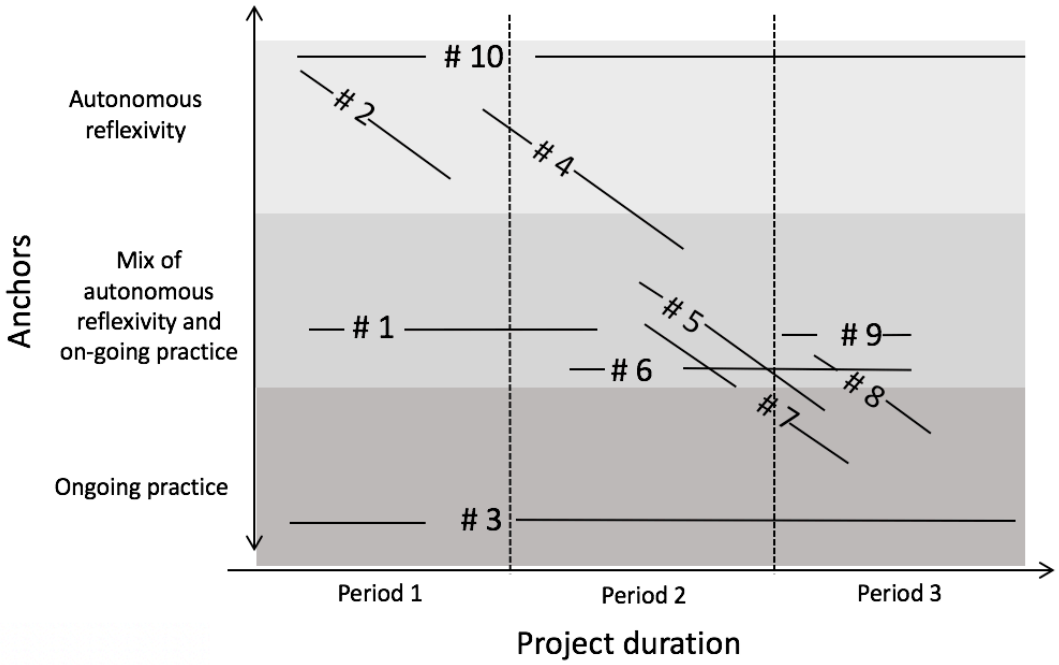

Figure 3. Unfolding Pattern of Improvisational Forms[1]

As Figure 3 shows, during the first period of the project, the improvisational forms "revising voluntarily" (#2) and "creating solutions spontaneously" (#10) were anchored to autonomous reflexivity. The former happened spontaneously during the design of the new graphical interface and other improvements, and the latter involved the disregarding of organizational practices. Thus, the triggers for autonomously reflexive improvisation were self-motivation and inspiration.

While the improvisational form "discovering and appending" (#4) occurred in the first period, it became progressively more anchored in ongoing practices in the second period in that it mixed autonomous reflexivity and ongoing practice. "Planning while executing" (#1) and "changing design practices and adapting to new situations" (#6) occurred in the first and second periods and became progressively less frequent with structural constraints. However, developers could still disregard some of these constraints and devise innovative solutions as Scott's temporary fix to the persistent PCB problem illustrates. Therefore, there were improvisational forms in the early and middle period of the project that self-motivation and inspiration initially triggered, but, as the project progressed, the triggers became more related to context.

Transitioning to the third and final period of the project, the improvisational forms "altering design features in response to shifting circumstances" (#5) and "channeling knowledge to respond to emergent challenges" (#7) became progressively more anchored in ongoing practices. Despite the increasing degree of structure, "channeling knowledge to respond to emergent challenges" (#7) continued for longer as the new product accommodated some of the changes. As per Figure 3, the improvisational form "devising responses to urgent needs" (#8) became more observable in these final stages of the project and more anchored in ongoing practices. Other forms, including "experimenting with work organization" (#9) and "appropriating known practices differently" (#3), were also anchored more in established practice (the latter was particularly based on ISO). These established practices involved the procedures of quality-management standards in the automobile industry and the routinized practices of the systems development project team. While these ISO-based practices and structures were present in the early and middle periods as well, the developers could improvise around them. The triggers for the forms of

[1] **Numbers #:** forms of improvisation (from Table 3): 1) planning while executing, 2) revising voluntarily, 3) appropriating known practices differently, 4) discovering and appending, 5) altering design features in response to shifting circumstances, 6) changing design practices and adapting to new situations, 7) channeling knowledge to respond to emerging challenges, 8) devising responses to urgent needs, 9) experimenting with work organization, 10) creating solutions spontaneously.
Lines: represent duration of the occurrence of the improvisational form within the project's duration.
Downward slope of lines: improvisational forms becoming less related to autonomous reflexivity and more anchored in ongoing practices.

improvisation associated with this later stage related more to the socio-technical context with developers' growing resistance to structural constraints shaping their efforts to find effective workarounds (e.g., the introduction of system integration testing in the project's third period).

In Table 4, we extend the discussion by deriving three "levels" of improvisation for the forms we discovered in the data: fluid (based on autonomous reflexitivity; see top of Table 4), anchored (based on the anchor of ongoing practice; see bottom of Table 4), or mélange (based on a mixture of the two; see middle of Table 4). We discuss this table and its implications in Section 5.

Table 4. Forms of Improvisation at ImproCo

Forms of improvisation	Occurrence and changes over time	Level of improvisation
Creating solutions spontaneously (#10)	From the start of the project and continued throughout; several occurrences.	Fluid improvisation
Revising voluntarily (#2)	At the start of the project but became more structured and ceased; many occurrences.	Fluid improvisation
Discovering and appending (#4)	In the early period but became more structured and ceased; fewer occurrences as project became more structured.	Fluid improvisation/ Improvisational mélange
Planning while executing (#1)	At the start of the project and continued for some time; many occurrences.	Improvisational mélange
Altering design features in response to shifting circumstances (#5)	Middle period of the project but continued before becoming more anchored and ceasing; many occurrences.	Improvisational mélange
Changing design practices and adapting to new situations (#6)	Middle period of the project and continued for some time; a few occurrences (compared to "Altering design in response to shifting circumstances" increased anchored improvisation).	Improvisational mélange
Experimenting with work organization (#9)	Late period of the project and continued for some time; few occurrences.	Improvisational mélange
Channeling knowledge to respond to emerging challenges (#7)	Middle stages of the project but continued before becoming more anchored and ceasing; fewer occurrences as it became more anchored in practice.	Anchored improvisation
Appropriating known practices differently (#3)	Throughout the project, anchored in ongoing practices; multiple occurrences, but fewer as the project progressed and new practices became established.	Anchored improvisation
Devising responses to urgent needs (#8)	Late period of the project but became more anchored and ceased; fewer occurrences as it became more anchored in practice.	Anchored improvisation

5 Discussion and Implications

In this section, by drawing on the patterns and forms of improvisation identified from the analysis we describe above, we address our research questions: that is, what kinds of improvisation exist in a new systems development project in the context of rigid quality-management standards and how these kinds of improvisation unfold over time. We then outline the research implications of these findings.

5.1 The Unfolding of Forms and Levels of Improvisation over Project Time

As we discuss Section 4, we identified ten major forms of improvisation at ImproCo and their triggers and their socio-technical context (Table 3). The triggers also reflected tensions of which we found three sets: 1) following formal plans of action while coping with emerging requirements, 2) allowing flexibility to change while stabilizing the system, and 3) finding imaginative workarounds while enforcing quality standards.

The improvisational responses were anchored in autonomous reflexivity, ongoing practices, or a combination of the two (Figure 3). Only three forms of improvisation were anchored in autonomous reflexivity; either ongoing practice or a hybrid of the two shaped the remaining forms of improvisation. We illustrate in Figure 3 how the different forms of improvisation followed a pattern. Table 4 summarizes the

unfolding forms of improvisation and groups them into three levels: fluid improvisation, anchored improvisation, and improvisational mélange. We elaborate on these improvisational levels below.

5.1.1 Fluid Improvisation

As we illustrate in our analysis, some forms of improvisation are more spontaneous (individuals' autonomous reflexivity) and often seen by others as an innovative and "bold" move (see Section 4.1). We call this level of improvisation "fluid improvisation" (see top of Table 4). This includes solo performances and the initial stages of some other forms of improvisation, such as revising voluntarily and discovering and appending. The triggers for these forms of improvisation are spontaneous in their nature because they are based on personal inspiration and serendipity, which Scott illustrated in the first period of the project by working on the weekend to find a solution. Improvisation in such cases demonstrated imagination (which the developers' ability to anticipate future requirements informed) and creativity (given the developers' ability to come up with a solution on most occasions).

5.1.2 Anchored Improvisation

At the bottom of Table 4, we show forms of improvisation that are more anchored in established, ongoing practice. These established practices are, in part, embedded in quality-management standards such as ISO 9001 and, at ImproCo, were in the schedules and routinized practices of the systems development project team, too. We call this level of improvisation "anchored improvisation", and it reflects associated structural influences. This form of improvisation includes appropriating previous practices differently and other forms of improvisation in the final stages (e.g., responding to urgent needs, responding to emergent challenges). The triggers for the forms of improvisation associated with this level were more systemic; they became stronger as the project became more developed and structured through repeated improvisational occurrences and increasing systemic influences such as stabilizing technology as per the case study's third period.

5.1.3 Improvisational Mélange

Between the fluid and anchored levels, we show a cluster of improvisational forms that often start "fluidly" but gradually become anchored in established practices in Table 4. We call this middle level "improvisational mélange". The triggers for these forms of improvisation were related to self-motivation and inspiration at the project's start but became more grounded in the socio-technical context. The developers frequently sought to improvise solutions outside their "normal" work routine (as Scott's work illustrates), which helped to expand the possibilities of the hardware and software under development. The amount of improvisational activity seemed to relate to various milestones throughout the project lifecycle. Hence, developers' and managers' experience of project time, represented by the pressure to meet certain milestones, helped to shape the improvisational forms, mainly at the improvisational mélange level. Anchored improvisation increased as developers were able to evoke prior experiences and to appropriate established practices. However, the fluid improvisational activities occurred less as they became more anchored in structuring practices.

As Table 4 shows, most of the improvisation forms were at either the improvisational mélange or anchored improvisation levels. This finding indicates that, while many of the forms of improvisation occurred later in the project, the earlier forms of improvisation were qualitatively different (e.g., they were based on autonomous reflexivity in contrast to the later ones that were anchored on structures such as ongoing practices and procedures) (Pavlou & El Sawy, 2010). In Section 5.2, we capture this phenomenon through the "paradox of progressive saturation".

5.2 The Changing Nature of Improvisation: A Paradox of Progressive Saturation

Our analysis indicates that, as a systems development project progresses, fluid improvisational work becomes increasingly saturated with structural influences such as ongoing practices and procedures. Yet, while development team members expect to work in a more structured way, their increasing knowledge about the project yields progressively more innovative forms of improvisation. For example, in our case study, we found that developers used unconventional means to handle last-minute customer requests despite looming ISO-based sign-offs.

At the beginning of the project, developers had greater latitude for autonomous reflexivity because improvisation was only loosely anchored in structural influences; it was more fluid and subjective. In the middle

and later periods of the project as the influence of structural constraints (e.g., structuring processes such as the sedimentation of routine practices and the stabilization of technology features) on the nature of improvisation became stronger, improvisation became more anchored in practices. And yet, the developers' increasing knowledge about the project yielded progressively more innovative forms of improvisation. The developers could, therefore, draw on their highly local and timely knowledge to improvise innovative solutions.

Finally, to illustrate the evolutionary process of improvisation and its constituent forms, we conceive of it as a paradox funnel (see Figure 4).

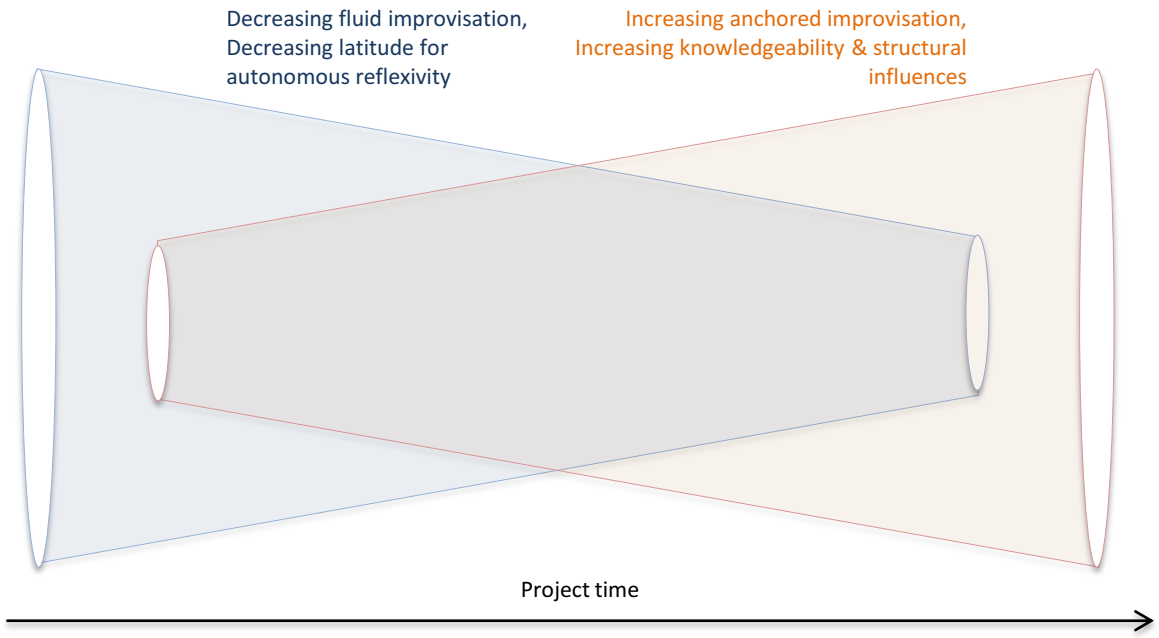

Figure 2. The Progressive Saturation Paradox Funnel

Moving from left to right, we depict that the influence of structural constraints on improvisation increasingly saturates the initial latitude for autonomous reflexivity and fluid improvisation. Yet, paradoxically, the increasing project knowledgeability of the developers yields progressively more innovative forms of anchored improvisation, which we call a "paradox of progressive saturation". It is paradoxical because the nature of improvisation over the course of a systems project reveals seemingly contradictory elements of the improvisation; that is, it simultaneously becomes both more saturated (through the increasing influence of structural constraints) and more dynamic (through developers' being progressively more innovative by drawing on their increasing knowledgeability). As new systems development projects progress, the nature of improvisation can transform from fluid forms of improvisation to forms of improvisation that are anchored more on ongoing practices and structures. This paradox funnel depicts the changing nature of improvisation over the course of a single systems project in the context of rigid quality-management standards.

This conceptualization of the changing nature of improvisation has several research implications. First, our characterization of the nature of improvisation as a paradox of progressive saturation (Figure 4) complements existing views of improvisation paradoxes in the IS literature, such as "planned serendipity" and "rehearsed spontaneity" (cf. Zheng et al., 2007), by adding another dimension to illustrate the tensions and dynamics that occur as different forms of improvisation unfold. The systems development literature often treats improvisation as homogeneous in the lifetime of a systems project. Research on improvisation in systems projects can benefit from insights into the dynamics of tensions, triggers, and their socio-technical context, their associated forms of improvisational practice, and how these improvisational forms change over the course of a systems development project lifecycle, such as the one we have empirically investigated.

Second, insights from the study also contribute to better explaining the paradoxical tensions between control and improvisational flexibility in systems projects (cf. Stacey & Nandhakumar, 2009) in the context of safety-critical, rigid quality-management standards. The literature on ISO 9001 suggests that the imperatives for formal procedures (Benner & Tushman, 2002) and bureaucracy (Singels, Ruël, & van de

Water, 2001) discourage the autonomous, "creative" thinking that we observed. However, in our case, ISO implementation actually forged improvisation due to the individual developers' ingenuity and their localized culture of experimentation. Their knowledgeability, in terms of project, domain, and technical knowledge, enabled them to enter into particular "levels" of improvisation (i.e., more autonomously reflexive or more anchored in practice); the projects earlier days provided more latitude for autonomous reflexivity and fluid improvisation. While the scope for fluid improvisation decreased towards the end of the project, the forms of improvisation became more innovative as the project ran out of time and structural influences took hold. We found that the latitude to improvise later on in the project related more to the accumulated local project knowledge and practices (including the practice of improvising), which higher degrees of fluid improvisation earlier on in the process balanced (creating solutions spontaneously, discovering and appending, and revising voluntarily). Therefore, accumulated knowledge was at a premium later on in the project if the developers were to pull off any form of improvisation at all, which they did.

Finally, our findings also contribute to understanding the dynamic interplay between improvisation and learning. According to Weick (1998, p. 546), "improvisation does not materialize out of thin air"—prior experiences are key. As the bespoke ImproCo project progressed, the knowledge and learning of the developers grew, which contributed to their producing increasingly innovative forms of improvisation. Further, the actors could reflect on progressive "lessons" through their capacity to disembed from routines and break away from structural constraints, which helped them to restructure the emergent structural constraints. In every moment of improvisation, one has an opportunity to push the boundaries of what is possible with the existing artifacts (Scarbrough, Panourgias, & Nandhakumar, 2015) and expertise and to create new accumulations of knowledge (knowledgeability). Different forms of such knowledge are associated with different levels of improvisation. For example, earlier in the project, the fluid level of improvisation depended less on localized knowledge and the project's structural settings and more on broader knowledge and personal inspiration. As Moorman and Miner (1998b, p. 703) claim, this level of improvisation discards links between existing practices and "composes new patterns". That being the case, anchored improvisation later in the project, which exploited previous practices, depended significantly more on accumulated local knowledge and the recursive occurrence of practices, which helped reproduce and legitimize the knowledge. Knowledge creation and increasing knowledgeability could be seen as an unintended outcome improvisational act. This finding is interesting when compared to Miner et al. (2001, p. 331), who argue that improvisation differs from other organizational learning, such as a planned set of procedures to explore and retain knowledge.

6 Conclusion

In this paper, we conceptualize the changing nature of improvisation over the course of a systems development project. Specifically, we identify ten major forms of improvised practice and their triggers and socio-technical contexts in the context of safety-critical, rigid quality-management standards. We argue that these forms of improvisation are anchored in autonomous reflexivity, ongoing practices, or a combination of the two, which results in three "levels" of improvisation: fluid improvisation, anchored improvisation, and improvisational mélange. We show how, as the project we examined progressed, the increasing saturation of structural influences on improvisation constrained developers' latitude to improvise. Yet, paradoxically, their increasing knowledge yielded progressively more innovative forms of improvisation, which we refer to as a "paradox of progressive saturation".

Our research comes with several practical implications. The findings that developers could achieve different forms of improvisation throughout the project (with varying success) despite reinforcing the rigid context of planning and quality-management indicates that management cannot solely rely on detailed planning to resolve complex development set-ups. Process-management frameworks (such as ISO 9001) generate an overemphasis on tools, techniques, and methodologies, but the prescribed procedures alone can rarely resolve all difficulties (cf. du Plooy, 2002). Improvisational activities help developers cope with such difficulties by maintaining flexibility. However, placing sole trust in the improvisational capabilities of developers and managers without planning might cause other problems, such as inconsistency, poor coordination, and quality problems. To retain an innovative capability, firms must maintain the right balance between the support structure (e.g., based on ISO framework) and the latitude for improvisation needed to overcome these problems.

Further, we show that the process-management framework of ISO 9001 was sometimes at odds with the improvisational practices at ImproCo because the company needed to maintain innovative capacity. Nevertheless, the developers saw the frameworks as essential for maintaining a perception of quality (cf.

Nandhakumar & Avison, 1999) and satisfying annual ISO 9001 assessments. Such use of process-management frameworks as "scaffolding" seems to allow, nurture, and contain some improvisations in the daily work of system development. At ImproCo, the developers and managers followed the process-management framework just enough to sustain the ISO certification process but left enough room to practice a variety of improvisations.

Our research also comes with some limitations. In particular, we examined only a single case study and focused on improvisation at the project level. Studies in multiple contexts may facilitate cross-comparisons to identify possible variation in how improvisation unfolds over time in other settings. Further, individual project team members' experiences seemed to differ, yet we captured this phenomenon only partially. Future research could seek to develop and analyze multiple, overlapping narratives (cf. Brown, Stacey, & Nandhakumar, 2008) of project members' experiences, which could help to build on this work to develop a richer perspective of how improvisational forms unfold over the course of a project.

References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: Review and analysis.* Espoo, Finland: VTT Publications.

Archer, M. S. (2007). *Making our way through the world*. Cambridge, UK: Cambridge University Press.

Bansler, J., & Havn, E. (2003). Improvisation in action: Making sense of IS development in organizations. In *Proceedings of the International Workshop on Action in Language, Organisations and Information Systems* (pp. 51-63).

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., & Slaughter, S. (2003). Is Internet-speed software development different? *IEEE Software*, *20*(6), 70-77.

Benner, M., & Tushman, M. (2002). Process management and technological innovation: A longitudinal study of the photography and paint industries. *Administrative Science Quarterly*, *47*(4), 676-706.

Berente, N., & Yoo, Y. (2012). Institutional contradictions and loose coupling: Postimplementation of NASA's enterprise information system. *Information Systems Research*, *23*(2), 376-396.

Bhardwaj, A., Bhattacherjee, S., Chavan, A., Deshpande, A., Elmore, A., Madden, S., & Parameswaran, A. (2015). DataHub: Collaborative data science & dataset version management at scale. In *Proceedings of the Biennial Conference on Innovative Data Systems Research.*

Brown, A. D., Stacey, P., & Nandhakumar, J. (2008). Making sense of sensemaking narratives. *Human Relations*, *61*(8), 1035-1062.

Cunha, M., Cunha, J., & Kamoche, K. (1999). Organizational improvisation: what, when, how and why. *International Journal of Management Reviews, 1*(3), 299-341.

du Plooy, N. (2002). Information systems as social systems. In Cano, J. (Ed.) *Critical reflections on information systems*. London, UK: Idea Group.

Dybå, T. (2000). Improvisation in small software organizations. *IEEE Software*, *17*(5), 82-87.

Dyer, W. G., & Wilkins, A. L. (1991). Better stories, not better constructs, to generate better theory. *Academy of Management Review*, *16*(3), 613-619.

Effah, J., & Abbeyquaye, G. (2014). How FOSS replaced proprietary software at a university: An improvisation perspective in a low-income country. *African Journal of Information Systems*, *6*(1), 9-25.

Gioia, T. (1988). *The imperfect art: Reflections on jazz and modern culture*. Stanford, CA: Stanford Alumni Association.

Goffman, E. (1959). *The presentation of self in everyday life*. Garden City, NY: Doubleday.

Karlström, D., & Runeson, P. (2005). Combining agile methods with stage-gate project management. *IEEE Software*, *22*(3), 43-49.

Langley, A. (1999). Strategies for theorizing from process data. *Academy of Management Review*, *24*(4), 691-710.

Lanzara, G. (1999). Between transient constructs and persistent structures: Designing systems in action. *Journal of Strategic Information Systems*, 8(4), 331-349.

Leidner, D. E. L., & Jarvenpaa, S. L. (1993). The information age confronts education: Case studies on electronic classrooms. *Information Systems Research*, 4(1), 24-54.

Louridas, P. (1999). Design as bricolage: Anthropology meets design thinking. *Design Studies*, *20*(6), 517–535.

Magni, M., Proserpio, L., Hoegl, M., & Provera, B. (2009). The role of team behavioral integration and cohesion in shaping individual improvisation. *Research Policy*, *38*(6), 1044-1053.

Marjanovic, O., & Hallikainen, P. (2013). Disaster recovery—new challenges and opportunities for business process management research and practice. *Pacific Asia Journal of the Association for Information Systems*, 5(1), 23-44.

McGann, S. T., & Lyytinen, K. (2008). The improvisation effect: A case study of user improvisation and its effects on information system evolution. In *Proceedings of the International Conference on Information Systems.*

Mendonça, D. (2007). Decision support for improvisation in response to extreme events: Learning from the response to the 2001 World Trade Center attack. *Decision Support Systems*, *43*(3), 952-967.

Miles, M., & Huberman, M. (1994). *Qualitative data analysis*. Thousand Oaks, CA: Sage.

Miner, A., Bassoff, P., & Moorman, C. (2001). Organizational improvisation and learning: A field study. *Administrative Science Quarterly*, *46*(2), 307-337.

Mirvis, P. (1998). Practice improvisation. *Organization Science*, *9*(5), 586-592.

Moorman, C., & Miner, A. (1998a). The convergence of planning and execution: Improvisation in new product development. *Journal of Marketing*, *62*(3), 1-20.

Moorman, C., & Miner, A. (1998b). Organizational improvisation and organizational memory. *Academy of Management Review*, *23*(4), 698-723.

Mutch, A. (2010). Technology, organization, and structure—a morphogenetic approach. *Organization Science*, *21*(2), 507-520.

Nandhakumar, J., & Avison, D. (1999). The fiction of methodological development: A field study of information systems development. *Information Technology & People*, *12*(2), 176-191.

Nandhakumar, J., & Jones, M. (1997). Too close for comfort? Distance and engagement in interpretive information systems research. *Information Systems Journal*, *7*(2), 109-131.

Nicolini, D. (2009). Zooming in and out: Studying practices by switching theoretical lenses and trailing connections. *Organization Studies*, *30*(12), 1391-1418.

Njenga, K., & Brown, I. (2012). Conceptualising improvisation in information systems security. *European Journal of Information Systems*, *21*(6), 592-607.

Orlikowski, W. J. (1993). CASE tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly*, *17*(3), 309-340.

Paulk, M. C., Curtis, W., Chrissis, M., & Weber, C. B. (1993). Capability maturity model, version 1.1. *IEEE Software*, *10*(4), 18-27.

Pavlou, P. A., & El Sawy, O. A. (2010). The “third hand”: IT-enabled competitive advantage in turbulence through improvisational capabilities. *Information Systems Research*, *21*(3), 443-471.

Rakitin, S. R. (2006). Coping with defective software in medical devices. *Computer*, *39*(4), 40-45.

Rodon, J., Sese, F., & Christiaanse, E. (2011). Exploring users' appropriation and post-implementation managerial intervention in the context of industry IOIS. *Information Systems Journal*, *21*(3), 223-248.

Saltz, J. S. (2015). The need for new processes, methodologies and tools to support big data teams and improve big data project effectiveness. In *Proceedings of IEEE International Conference on Big Data* (pp. 2066-2071).

Scarbrough, H., Panourgias, N. S., & Nandhakumar, J. (2015). Developing a relational view of the organizing role of objects: A study of the innovation process in computer games. *Organization Studies*, *36*(2), 197-220.

Schrenker, R. A. (2006). Software engineering for future healthcare and clinical systems. Computer, 39, 26–32.

Singels, J., Ruël, G., & van de Water, H. (2001). ISO 9000 series—certification and performance. *International Journal of Quality and Reliability Management*, *18*(1), 62-75.

Stacey, P., & Nandhakumar, J. (2009). A temporal perspective of the computer game development process. *Information Systems Journal*, *19*, 479-497.

Stacey, P., & Nandhakumar, J. (2008). Opening up to agile games development. *Communications of the ACM*, *51*(12), 143-146.

Teoh, S. Y., Wickramsinghe, N., & Pan, S. L. (2012). A bricolage perspective on healthcare information systems design: An improvisation model. *ACM SIGMIS Database*, *43*(3), 47-61.

Walsham, G. (1993). *Interpreting information systems in organizations*. Chichester, UK: Wiley.

Weick, K. (1993). The collapse of sensemaking in organizations: The Mann Gulch disaster. *Administrative Science Quarterly*, *38*(4), 628-652.

Weick, K. (1998). Improvisation as a mindset for organizational analysis. *Organization Science*, *9*(5), 543-555.

Weick, K. E. (1999). That's moving: Theories that matter. *Journal of Management Inquiry*, *8*(2), 134-142.

Weick, K. (2001). *Making sense of the organization.* Maldon, MA: Blackwell Publishing.

Zheng, Y., Venters, W., & Cornford, T. (2007). *Distributed development and scaled agility: Improvising a grid for particle physics*. Retrieved from http://is2.lse.ac.uk/wp/pdf/wp163.pdf

Zheng, Y., Venters, W., & Cornford, T. (2011). Collective agility, paradox and organizational improvisation: The development of a particle physics grid. *Information Systems Journal*, *21*(4), 303-333.

Appendix

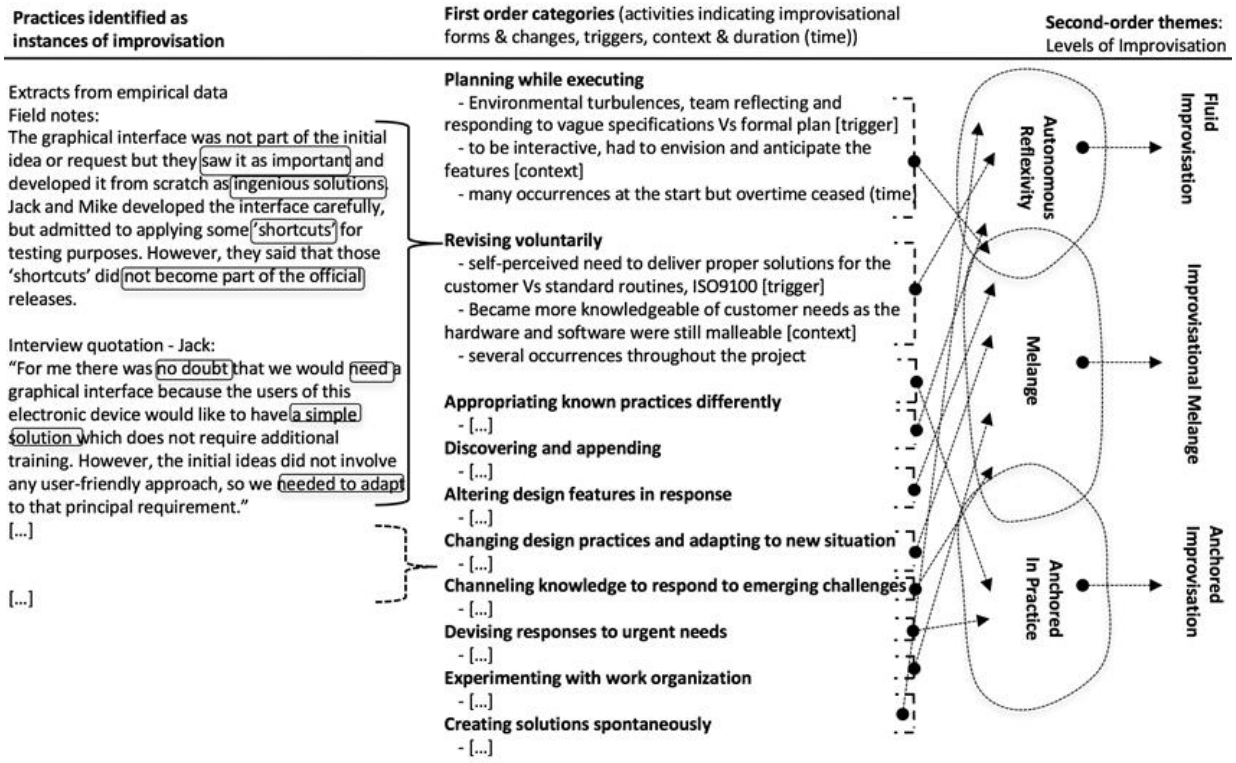

Figure A1. Snapshot of the Evolution Analysis Process

About the Authors

Wolfgang A. Molnar works as a researcher at Warwick Business School and practitioner in the automotive industry for ZF in Germany. Previously, he was a Research Fellow at the Luxembourg Institute of Science and Technology. He earned his PhD in Information Systems and Management from the Warwick Business School, University of Warwick. His research expertise focuses on socio-technical aspects in developing Information Systems.

Joe Nandhakumar is Professor of Information Systems at Warwick Business School, University of Warwick. He earned his PhD from the University of Cambridge, Department of Engineering. His primary research and teaching interests focus on the digital innovation and organizational and societal transformation. His recent work has been published in the journals such *as Information System Research*, *Journal of the Association for Information Systems, Information Systems Journal* and *Organization Studies*.

Patrick Stacey is an Associate Professor of Information Management at the School of Business and Economics, Loughborough University. His expertise lies in managing games development and a variety of epi-phenomena that stem from this, most particularly emotion and improvisation. Over the course of his 25-year career in industry and academia, he has won a number of prestigious prizes, fellowships and scholarships from institutions including the Association for Computing Machinery, Engineering and Physical Sciences Research Council, Imperial College London, Warwick Business School, and the National Computer Board of Singapore. The varied positions he currently holds includes: Senior Editor of *IT and People*, Program Director for Masters in Information Management and Business Technology, Academic Lead on the Digital and Health Helix for EU Vision2020 and Research Challenge Advocate for Enabling Technologies.