JITTA

JOURNAL OF INFORMATION TECHNOLOGY THEORY AND APPLICATION

A Publication of the Association for Information Systems

A Model for Analyzing Changes in Systems Development Practices

Even Åby Larsen
Department of Information Systems
University of Agder, Kristiansand, Norway
Even.Larsen@uia.no

Tero Päivärinta
Computer and Systems Science
Luleå University of Technology, Sweden
& Department of Information Systems
University of Agder, Kristiansand, Norway
Tero.Paivarinta@ltu.se

Kari Smolander
Department of Information Technology
Lappeenranta University of Technology, Finland
Kari.Smolander@lut.fi

Abstract:

This paper introduces an empirically grounded model for analyzing intended and unintended changes in the prevalence of information systems development practices. In the model, any development practice observable in a development organization can be analyzed according to two dimensions: the intended scope of defined practices versus the actual scope of enacted practices.

Furthermore, the model identifies eight types of change paths in systems development practices based on the two dimensions: emergence, entropy, initiation, abandonment, formalization, informalization, implementation, and recalcitrance. The eight types of change paths provide an integrated theoretical model for understanding how systems development practices can change in organizations and projects and among individual developers in a given context. The paper concludes by discussing how the model complements and integrates concepts of the contemporary research on systems development practices and outlines its potential uses for future research.

Keywords: systems development, software development, practice, change

Volume 13, Issue 3, pp. 21-48, September 2012

Karlheinz Kautz was the Senior Editor for this paper.

A Model for Analyzing Changes in Systems Development Practices

INTRODUCTION

Information Systems (IS) scholars have studied systems development methods and practices for many years. From early on, systems development research has focused on how development should be carried out (e.g., Dijkstra 1965; Parnas 1972). Through the 1970s and 1980s, this led to the construction of numerous systems development methods (SDMs; Jayaratna 1994; Avison and Fitzgerald 2003). Adherence to methods was regarded by default as useful; methods were to be rigorously predefined and utilized as intended by the method developers (Humphrey et al. 1991; Jarke et al. 1994). More recently, the idea of using predefined development methods and practices has widened to software process improvement (ISO/IEC 2002, Software Engineering Institute 2006) and to the standardization of development processes (ISO/IEC 2008).

Despite the vast focus on methods, many of them have a limited conceptual and empirical foundation without extensive scientific evidence (Fitzgerald et al. 2003). Several methods may have been constructed based on the intuition and experience of those who participated in their creation, or they are based on weakly grounded “best practices” or conventional wisdom. Also, large-scale studies of method use and adoption for information systems development (ISD) are rare. Fitzgerald (1998a) reported that 60 percent of development organizations did not use any methods. Baskerville et al. (2003) point out that the high speed development of the Internet age has led developers to abandon traditional engineering principles, perhaps indicating that those principles are not universal. Kautz et al. (2007) and Siau and Rossi (2011) highlight that more research is needed to fully understand the impact of implementing different practices.

Moreover, in contrast to the ideal of adhering to predefined methods and practices, researchers have noted that organizational practices may be evolutionary (Kostova and Roth 2002), they may involve a tacit component (Kogut and Zander 1992), and they are performed uniquely with variation in a certain time and place (Pentland and Feldman 2005). Regardless of whether predefined SDMs are adopted or not, systems developers and development organizations thus have a number of varying practices, such as change management, testing, and requirements elicitation practices, that the developers and organizations use repeatedly. The developers and organizations may have agreed on them, the organizational management may have instructed the organization to use certain practices, or the practices may have been shaped and formed by the history and the experiences of the developers and the organization. Some practices may be standardized and some may be documented, whereas others are less explicitly defined or documented but still repeatedly used.

Being motivated by this unclear nature of practices and by the continuing scarcity of in-depth empirical research on actual development practices and methods-in-use (Siau and Rossi 2011), we selected our research question: How are systems development practices formed in professional systems and software development organizations? Especially we are interested in how practices are taken into use, whether they remain in use or not, and what are their possible changes during their lifetimes. We gathered data about local systems development practices in eleven systems and software development organizations and built a theoretical model that can explain practice formation and change.

CONTRIBUTION

The paper presents a model for describing the intended scope versus the actual scope of information systems development (ISD) practices and demonstrates how the model can be used to analyze changes in the scope of the practices. The model proposes eight stereotypical change paths of development practices in an organization: emergence, entropy, initiation, abandonment, formalization, informalization, implementation, and recalcitrance.

We illustrate the value of the change path model for researchers by using it as an analytical tool for understanding and integrating hitherto scattered theorizing about systems development practices and change in them. The paper analyzes the concepts of previous theories of analysis of changes in systems development practice, such as “method-in-action” (Fitzgerald et al. 2002), “amethodical systems development” (Truex et al. 2000), and the Capability Maturity Model (CMM) for software process improvement (Humphrey 1989) in the light of the suggested model. The model also complements the previously used theory of diffusion of innovation (DoI; Rogers 1995; Mustonen-Ollila and Lyytinen 2003) to explain change in ISD practices. All in all, the model integrates previous theories and discussions about planned versus emergent practices into a joint frame of reference, which also includes the phenomena of recalcitrance, abandonment, and entropy of practices. It can be used as an analytical tool in future research on how and why systems development practices change in development organizations. For practitioners, the change path model can help to understand the status and change pressures of and management needs for local practice repertoires.

The first part of the study, instructed by the Grounded Theory research approach (Glaser and Strauss 1967; Strauss and Corbin 1990), resulted in a model of ISD practices. The model involves two dimensions of systems development practices that can vary independently: the intended scope and the actual scope of enactment. In the second part we analyzed changes in practices and recognized pressures to change them in light of these dimensions. As a result, we identified eight types of change paths of systems development practices: emergence, entropy, initiation, abandonment, formalization, informalization, implementation, and recalcitrance. We argue that the model with the eight change path types integrates and complements previous research on systems development practices. Moreover, we argue for the future usefulness of the model for forming more exact theories explaining why particular types of practices may be differently shaped across development organizations and what would be the consequences of variance in particular practices on development performance and results.

After providing a conceptual and theoretical background for our study in Section 2, Section 3 describes the research process in more detail. The model of analysis of the intended and actual scopes of practice enactment is illustrated in Section 4, and its use in identifying the stereotypical change paths of systems development practices is then presented in Section 5. Section 6 discusses the previous concepts related to development practices in the light of the construct, suggests some uses of the results both in practice and in research, and notes some enhancements that could be made to the study in the future. Section 7 concludes the paper.

CONCEPTUAL BACKGROUND AND RELATED RESEARCH

Essential concepts

We targeted our study at organizations which practiced *software development*. That is, they were “concerned with creating descriptions of the purposes of the software, of its problem domain, of its structure and behavior, of the computations to be performed, of the interfaces between the software and its environment and its users …” and with implementing the resulting software product in a computer system (e.g., Jackson 1992, p. 34). Moreover, all our target organizations also practiced *information systems development*, being involved in change processes “taken with respect to object systems in a set of environments by a development group to achieve and/or to maintain some [organizational] objectives” (Lyytinen 1986, p. 74; parenthetical content added by the authors). That is, we were interested in systems and software development organizations, which created and maintained software systems or products to be used by an organization or organizations for their business purposes. The early idea of distinguishing between system and software requirements (e.g., Royce 1970) highlights the difference between these terms. However, from here on we use the term *systems development* to highlight that both system and software development are essentially included in the whole phenomenon of our interest and that our target development organizations need to master their profession with regard to both of these fields.[1]

It is important to declare what we mean by the concepts of “method” and “a practice.” A *systems development method* (*SDM)* can be defined as an organized collection of concepts, beliefs, values, and normative principles supported by material resources (Lyytinen 1987) which guide the development of information systems. All thorough methods define a model for the development process, a set of description or coordination techniques to be used in the process steps, a concept structure to be used in the description or coordination tasks, division of work by stakeholder roles, and a set of values which guide the work (Tolvanen 1998). A method may be supported by a set of tools supporting the descriptions and notations and the execution of the process. It can be targeted at a specific application domain, and it may involve a certain perspective on values and beliefs to guide the development process (Iivari et al. 1998, 1999).

While a method represents a thoroughly defined structure for development work, if implemented in a development organization, systems development may also include more loosely organized practices. Therefore, we use the concept of *a practice* as the key concept in this paper when we refer to both more and less organized and situated systems development activities that are conducted recurrently by human agents (Orlikowski 2002). Thus according to our definition, practices may include *both* thoroughly organized use of SDMs *and* loosely organized recurring activities that may use individual tools or techniques in systems development.

In addition to work practices of individuals and focal groups, which have been in the focus of a great many studies (Orlikowski 2002), a systems development practice may become an *organizational practice*, which can be defined as the organization’s routine use of knowledge. Organizational practices can exist at multiple levels, such as in a development project or in an organization taking part in several projects or development processes (e.g., systems

[1] A development organization making fully embedded software in automated technological environments without any direct human interaction with such software represents a subset of software development organizations. In our study, however, all of our target organizations needed to consider an organizational context in which their software products were implemented.

development teams, groups, or companies). Organizational practices often have tacit components embedded partly in individual skills and partly in collaborative social arrangements (Kogut and Zander 1992; Nelson and Winter 1982; Szulanski 1996).

There are different viewpoints on how organizational practices take shape. For example, Szulanski (1996) argues that a "best practice" represents organizational knowledge which can be transferred between a source and a recipient unit inside an organization as a replication of an organizational routine. On the other hand, Kostova and Roth (2002) suggest that an organizational practice "evolves over time under the influence of the organization's history, people, interests, and actions" and that it comes "to reflect the shared knowledge of the organization and tend[s] to be accepted and approved by the organizational members." That is, a practice may be rationally adopted or may emerge in an evolutionary manner in an organization. Within the latter view, any identifiable practice can be phrased to be meaningful to the extent to which it is "useful" in a contextual, situated organizational activity (Orlikowski 2002). We adopt the view of Orlikowski (2002) that it is useful to study how collective competence in organizations is enacted through a repertoire of identifiable organizational practices.

If we compare a systems development method with a practice, a method, if adopted in an organization, always embodies a set of predefined and idealized practices, whereas a practice is not always defined at the level of a method, at least with regard to all potential elements (Tolvanen 1998) of method knowledge. A practice can be explicitly defined, for example, as a technique specified by a method or can emerge as a habitual response to a recurrent situation (cf. Kostova and Roth 2002). In the former case, there exists an explicit definition of the practice in question and the organization has somehow expressed its importance.

Any existing description of a practice implies that at least one stakeholder in the organization has intended that it should be enacted. However, ethnographical studies since the 1980s have shown that such canonical practices often deviate from the actual actions taken (Brown and Duguid 1991). Moreover, systems development organizations may, to an extent, follow undocumented practices, as will be shown in this paper. In line with this, Pentland and Feldman (2005) highlight a distinction between the performative and ostensive aspects of organizational routines. The performative aspect represents "the specific actions taken by specific people at specific times when they are engaged in what they think of as an organizational routine" (p. 796). The ostensive aspect is "the abstract or generalized pattern of the routine" (p. 796). The ostensive aspect is not necessarily a formal written description of the practice; it can also be the response given by a developer when asked how he or she is carrying out his or her work.

During the empirical part of our study, we noticed quickly that not all practices were defined in a documented form, and, on the other hand, not all defined practices were enacted by the intended part of the target organizations. This led us to make an analytical distinction between the *intended scope* of a predefined practice and the scope of its actual *enactment*, that is, the extent of the actual enactment of the practice in question. An earlier inspiration for this view can be found in, for example, Argyris and Schön's theory of action (1974) and their influential concepts of espoused theory and theories-in-use. Theory-in-use is the implicit theory held by a human "that actually governs his actions," whereas espoused theory involves assumptions about more general or socially acceptable expectations concerning human behavior in the given situation (Argyris and Schön 1974, pp. 6–7). However, our concepts focus perhaps less on the fit between "inner feelings" of the stakeholders and their espoused theories of action, which is the focus of Argyris and Schön. Rather, our work plainly acknowledges that practices-in-use do not always follow the predefined, espoused, ideas of practices.

Related research: ISD methods versus emergent practices

The 1980s were the heyday of SDMs. The era introduced a multitude of methods created by famous consultants. The methods based on functional decomposition (Stevens et al. 1974; Constantine and Yourdon 1979; Gane and Sarson 1979; Lundeberg et al. 1981; Martin and Finkelstein 1981; Yourdon 1989) that were prevalent in the early 1980s had to make room for object-oriented analysis and design methods in the beginning of the 1990s (Booch 1991; Wirfs-Brock and Wilkerson 1989; Blaha et al. 1988; Coad and Yourdon 1990, 1991; Jacobson 1992). All these methods could be understood as commercial products with consultation opportunities. An organization could select an SDM and put it into use with minor modifications. These ideas culminated in the suggestions of contingency approaches to method selection in development organizations (Davis 1982; Iivari 1989) and their critique (Avison and Wood-Harper 1991).

In the 1980s, the "professional work practice" approach (Iivari and Lyytinen 1998), also called "reflective systems development" (Mathiassen 1998), was one of the first research programs which challenged the belief that development methods, as such, would improve the effectiveness of systems development (Iivari and Lyytinen 1998). The approach was developed through action research pursuing local improvements of development practice based on contextual circumstances (Mathiassen 1998). In the 1990s, the idea of situational and incremental, while simultaneously tool-focused and formal, method engineering in the context of development organizations started to gain ground (Kumar and Welke 1992). Later on, a number of systems development researchers have argued that methods undergo extensive pragmatic adaptation and emergence of unexpected issues during their adoption and use (Stolterman 1992; Introna and Whitley 1997; Fitzgerald 1998a, 1998b, 2003; Kautz 2004; Kautz et al. 2004; Vogelsang and Kensing 2006; Päivärinta et al. 2010), while there exist varying, even contradictory, rationales and pressures both for and against method use (Fitzgerald 1998b). It was even suggested that "amethodical" systems development represents an alternative view on development practice in contemporary development organizations. Ultimately, methods could have no prescribed role at all, while actual systems development practice would emerge through contextual interaction and improvisation (Truex et al. 2000).

However, despite considerable efforts and interest in developing more or less generic SDMs and other development process innovations, no systematic research tradition to understand their adoption and utilization in practice followed (Mustonen-Ollila and Lyytinen 2003). Moreover, previous studies have been limited with regard to the number of adoption decisions studied, types of development process innovations covered, or types of factors included in the study (Mustonen-Ollila and Lyytinen 2003). Mustonen-Ollila and Lyytinen (2003) approach development practices from the viewpoint of "process innovations" and view practice enactment through the theory of diffusion of innovations (DoI) (Rogers 1995). DoI distinguishes between initiation (the decision to use an innovation) and implementation (putting an innovation into use), and, therefore, it considers practice adoption as a deliberate and at least partly planned process. The theory identifies three types of adoption decisions: optional innovation decision, which is a personal choice made by an individual; collective innovation decision, which is made by a consensus within the organization; and authority innovation decision, which is made by few individuals who hold positions that give them the authority to make decisions. In this light, DoI makes a distinction between different organizational levels of adoption, but still sees adoption only as a rational and planned process. Our view is that, in this sense, DoI does not differ much from the traditional view of SDMs where methods are selected and put into use with rational decision making.

Table 1: Summary of Related Research

Unit of analysis/ Focus	**Individual**	**Project**	**Organization**
Normative pre-defined/ engineered methods	Developer use of and intentions to use methods (Stolterman 1992; Fitzgerald 1997; Hardgrave et al. 2003)	Project-specific method tailoring or engineering (Fitzgerald et al. 2003; Bajec et al. 2007)	Contingency approaches to method selection (Davis 1982; Iivari 1989), Method engineering (Kumar and Welke 1992) incrementally in development organizations (Tolvanen 1998), Variation in use of structured methods (Bansler and Bødker 1993), Organization-specific method precustomization (Fitzgerald et al. 2003, 2006)
Emergence/ contextual enactment of methods and practices	Educating reflective systems developers (Mathiassen and Purao 2002)	Local, project-specific method emergence and practice adoption (Fitzgerald et al. 2002; Kautz 2004; Madsen et al. 2006)	Lack of method use, emergence of practices (Whitley 1998), Project-by-project variation in method use (Kautz et al. 2004), Contextual emergence of agile methods-in-use (Vidgen and Wang 2009), Longitudinal change and emergence of methods-in-use in organizations (Mathiassen and Vogelsang 2005; Rowlands 2008; Päivärinta et al. 2010)

Contemporary research on development practices has studied the phenomenon on at least four levels of analysis: the individual systems developer, the development project, the development organization, and the industry. Below, we omit the industry-level analysis of development practices from our in-depth research agenda. We plainly notice that industry-level requirements, standards, and other professional recommendations may have a prescriptive impact on some development organizations. They may become one incentive among many to adopt methods in certain development organizations (Fitzgerald 1998b; Fitzgerald et al. 2003). Table 1 summarizes our discussion of the related research in relation to the identified levels of analysis.

A few empirical studies have focused on the viewpoint of individual systems developers and how they perceive the relevance of methods and utilize them in their work. These include works on the relationship between developer experience and modes of SDM use (Stolterman 1992, Fitzgerald 1997) or determinants of developer intentions to use methods (Hardgrave et al. 2003). Education of reflective systems developers with regard to SDMs has also received attention (Mathiassen and Purao 2002). A few studies have discussed development project exigencies in relation to local method adoption, where the unit of analysis has been a project (e.g., Fitzgerald et al. 2002; Kautz 2004; Madsen et al. 2006). While several researchers in the 1980s and early 1990s searched for contingency-factor theories to aid in a project's method selection, incremental method engineering was proposed later on as a more feasible approach to project-specific method creation (Fitzgerald et al. 2003; Bajec et al. 2007). However, despite the general-level observations on emergence of project specific methods-in-use (Fitzgerald 1998b; Fitzgerald et al. 2002), the lack of empirical evaluations of methods in practical settings continues (Siau and Rossi 2011).

A stream of research has discussed methods in the context of systems development companies and organizations beyond individuals and particular projects. In addition to the above-mentioned literature, for example, Bansler and Bødker (1993) examined variations in the use of structured analysis methods in three systems development organizations, and Tolvanen (1998) and Fitzgerald et al. (2003, 2006) focused on incremental engineering and pre-tailoring of software development methods in IT industry organizations. Whitley (1998) studied how pre-understanding guides practice formation and (lack of) method use in three prominent Web-design organizations. Kautz et al. (2004) shed light on reasons for project-by-project variation in method utilization in a software company, whereas Vidgen and Wang (2009) revealed contextual issues and emergent capabilities of teams that impacted on agility in the adoption of agile software development processes in two companies. Mathiassen and Vogelsang (2005) and Päivärinta et al. (2010) followed longitudinally how particular development methods have been adopted and adapted in professional development organizations. Rowlands (2008) discussed how institutional and social structures shaped the enactment of SDM in an Australian bank.

From the list of studies above, we see that systems development methods and practices have been approached on the one hand from the traditional, normative perspective, where they were seen as explicit and predefined instructions on how development should happen. The other perspective, which gained ground later on, sees practices and methods-in-use (cf. Fitzgerald et al. 2002 and Madsen et al. 2006) as evolutionary, partly tacit, situational, and formed through organizational history and experiences; that is, the practices are at least partly emergent (Table 1). We are lacking theories and models that could combine these two widely spread research perspectives in the ISD field and explain how systems development practices come into use, regardless of whether they are emergent or planned, and how they develop, erode, and disappear. Our research aims to fill this gap in the research and to provide an integrated model explaining how systems development practices evolve.

RESEARCH PROCESS

To understand systems development practices in different kinds of software development organizations, we selected *Grounded Theory* as the initial research approach. Grounded Theory is a research method developed originally for the social sciences by Glaser and Strauss in the 1960s (1967) and later developed and reinterpreted by the original authors (e.g., Strauss and Corbin 1990; Glaser 1978) and others (e.g., Martin and Turner 1986; Locke 2001). The approach has also been used to a fair extent in the IS discipline in varying forms (Urquhart 2007).

An essential principle in the research process has been the triangulation of data sources, investigators, and theories (Denzin 1978). The data were collected from eleven organizations and interviews were organized in three countries. The data analysis was done independently by three investigators, and then the results were combined after long discussions in several common workshops. In addition, we discussed and validated the analysis results in the light of several theories, such as diffusion of innovation (DoI; Rogers 1995), method-in-action (Fitzgerald et al. 2003), amethodical systems development (Truex et al. 2000), and software process improvement (Humphrey 1989).

In the following, we describe our data collection and analysis process in more detail.

Data collection

The empirical part of our study is based on forty-seven interviews and discussions with a variety of stakeholders related to systems development. We decided to concentrate on those roles that were directly related to either executing or managing the daily systems development work. These roles included developers, project managers, development managers, analysts, architects, and testing managers in eleven systems development organizations (Table 2). We decided to concentrate on systems development organizations only and excluded external roles, such as customers, from the list of stakeholder roles. The interviews were open-ended and focused on clarifying the development practices followed in those organizations through the following main themes:

- background information about development projects and histories typical for the target organizations
- whether any particular methods or practices had been followed in the projects, and brief descriptions of those
- whether some systems development related practices were followed in the whole organization, beyond the particular projects discussed
- the rationale for following a particular method or practice, if known by the interviewee

Table 2: Target Organizations

	Number of Employees (locations)	Business	Interviewees (length of work experience, in years)
A	6, one site	Small Web-based e-business solutions	Senior Developer (13)
B	8 (Norway, one site) + 10 (India)	Logistics solution for electricity networks	Developer (5) Development Manager ("well-experienced")
C	42, one site	System development and administration	Developer (8) Project Manager (30)
D	25 (Norway, one site), (Serbia)	Systems development and consulting	8 Developers and Project Managers (2–13)
E	4 (local office), 184 (in total, various sites in Norway)	Systems development and consulting	Developer/Consultant (8) Developer/Consultant/Project Manager (8)
F	800 (in multiple countries)	Operative systems for industry	3 Upper Managers (11, 20, and 22) 2 Project Managers (5 and 17) 2 Developers (7 and 12)
G	100 (Finland, one site)	Automation systems for industry	5 Upper Managers (8–20) 2 Project Managers (6 and 10) 1 Developer (6)
H	200 (Russia, one site)	Software development services, subcontracting	2 Upper Managers (10 and 26) 1 Project Manager (10) 2 Developers (7 and 10)
I	60 (Finland, one site)	Systems for a specific kind of resource planning	3 Upper Managers (4, 6, and 6) 2 Developers (5 and 5)
J	80 (Finland, one site)	Internal systems development organization	1 Upper Manager (20) 2 Project Managers (8 and 20) 1 Analyst (5) 1 Architect (5)
K	(a) 7 (in-house development and support, Norway, 2 sites) (b) 6 (Intranet development, several sites, + consultants)	Internal IT/IS development and support organization for a global corporation	(a) 1 Development Manager (40) (b) 1 Test Manager (10)

The interviewees were selected based on two criteria in addition to their accessibility: they needed to represent stakeholders who were related to systems development practice, and they needed to have long-enough work experience to have some historical insight about how the current practices in the target organizations had evolved. Typical roles of the interviewees included various roles of managers and developers. This triangulation of data (Denzin 1978) from different stakeholders can be considered a benefit from the viewpoint of Grounded Theory, where the purpose is to understand and explain theoretically the selected phenomenon in its rich real-life context.

The interviews typically lasted from thirty minutes to one hour, and they were tape-recorded and transcribed. The context of interviews in cases A–E resides in two Master's theses supervised by two of the authors (Nilsen 2008; Päivärinta et al. 2009) which focused on Norwegian systems development practices in small- and medium-sized enterprises. The researchers instructed the Master's students to collect data about the themes mentioned, and the data are used with permission of the respective Master's students. The interviews took place in Norwegian. In cases F–J, which took place after the preliminary case studies in Norway, the data were collected in relation to a Finnish research project led by one of the authors, and the interviews were conducted in Finnish. In Case K the data collection was conducted by one of the authors, in Norwegian, as a part of his general-level research cooperation with the organization in question. One member of the research team is a native Norwegian-speaker, whereas two members are native Finnish-speakers, and one of them also has fluent Norwegian skills. This made it possible to cross-check and discuss codes, which were documented in English.

Data analysis

The Grounded Theory analysis started in the inductive fashion due to a lack of unified theories and frameworks and due to our hopes of identifying new issues in relation to the previously identified issues and factors related to method adoption (such as Bansler and Bødker 1993; Khalifa and Verner 2000; Hardgrave et al. 2003). This was also our rationale for collecting data through open-ended interview, where only the list of the general-level themes (see above) guided the interview situation. The first concepts of our analysis resulted from open and axial coding and emerged gradually during discussions between the researchers. Finally, the resulting core construct of the change paths resulted from the research team's abductive reasoning (cf. Reichertz 2007) based on the hitherto defined concepts. These constructs reflected, in turn, back onto the collected data through selective coding. Data from Case K were collected later on in the selective coding phase to confirm the potential existence of the abducted change path categories.

In the open coding phase, we found, naturally enough, based on our interview themes, numerous mentions of *practices* related to different parts, phases, and styles of systems development processes, as well as varying *reasons* that seemed to lead to adoption of varying practices or lack of practices. After a while, the coding started to generalize toward an observation of varying *scope*s within which practices were defined (*project*, *organization*, sometimes *individual*) and that *enactment of practice*s could vary from the *normative scope of the defined practice*s (if any). We named the categories of possible local perceptions of the prevalence of the state-of-the-art of particular practices as either non-defined, individual, project-level, or organizational. These codes led us to establish the model of the NIPO grid (The definition is given below; see also Figure 1) on which we could locate particular types of practices in a particular context. After this observation, we also noticed from the data that several interviewees mentioned *change*s which had happened in their practices over time and *pressure*s to either *define* new practices or to *adjust* or even *abandon* certain practices. This led us forward to study the typology of how practices can change over time and which kinds of pressures may lead to which kinds of changes in the light of the NIPO grid.

The following excerpt illustrates our coding:

> *We knew about a tool, Systems Engineer … . It was used in one project, but it had the same weakness [as they had experienced with another case tool, Excelerator, in a previous project] in coming over to [maintain] the conceptual model of the actually realized system … . [The model of the system and the actual system] did not correspond to each other. That was the last time we used a CASE-tool.* (Development Manager, Case K)

This citation was coded to indicate: (a) a *project-specific practice*/tool (Systems Engineer, Excelerator) for modeling and (b) a *pressure* (incompatibility with the resulting models vs. actual systems) to abandon a project-specific practice/tool for modeling.

In axial coding we wanted to understand the connections between the levels of enactment and scope of identified systems development practices (see Figure 1, Appendix 1). We continued the analysis by asking whether we could identify general patterns of change pressures and changes in practices over time using the NIPO coordinates. In each case, we identified several such pressures (Appendix 2). However, there was too much variance in the adopted practices and contextual pressures having an impact on them to draw meaningful conclusions regarding detailed comparisons between the organizations. The results of analysis of the cases were all quite different from each other with regard to the practice repertoires revealed and the identified pressures to adopt, adjust, or abandon the practices. Each case had its own history and circumstances that had led to its own set of key practices and their change potential. In the light of the NIPO grid, we were now able to recognize that there were theoretically eight potential change directions (up–down, down–up, left–right, and right–left, each either toward or away from the grid's diagonal), which were already supported to a good extent by the existing organization-specific NIPO charts. We went through the data again and analyzed pressures against the intentions and enactments of practices and the directions of the pressures on the NIPO grid. We consider this analysis phase as *selective coding* (Strauss and Corbin 1990), because we were no longer looking for new theoretical categories and connections between them. Instead, the analysis required us to move back and forth between the data and theoretical constructs, and most of the analysis happened at theoretical level, which we were constantly able to ground with the observations from the data. The core category in this abductive stage emerged as "the change paths," and this meant that in the analysis we were able to describe the change in light of the theoretical constructs from the NIPO grid and simultaneously find evidence of such change paths of practices from the cases (Appendix 2).

We found researcher team meetings very productive when analyzing the data. The NIPO grid is the direct result of a long discussion in which we discovered that the scope of a practice actually had two dimensions. The change path concept emerged in a similar way after discussing unfruitful attempts at selective coding. This is an example of investigator triangulation (Denzin 1978), in which previously independent analyses by multiple investigators are combined into one analysis.

The analysis taught us that we would need more data to establish better theories of *why* particular practices (e.g., analysis, modeling, design, programming, testing) change over time and what the pressures are to change a practice. Hence, we can state that our research is still at its beginning. However, already this effort of studying eleven organizations was able to reveal more generic patterns of how practices can appear at a particular moment of time and how they may change over time due to varying pressures. These results provided us with the model of the NIPO grid and the eight change paths of development practices. Furthermore, we noticed that it is possible to explain, in light of the previous related literature, how our results integrate the previous research and provide improved understanding for further studies aimed at developing more concrete theories on systems and software development practices.

THE PRACTICE CONCEPT AND THE NIPO GRID

The practice concept emerged during open coding, when we realized that the interviewees spoke much more about their local practices than about general-level ISD methods. All interviewees mentioned and named actions that were carried out on a regular basis. Our analysis revealed that the practice category was connected to many other categories and that practice is actually a multidimensional concept. Of particular interest to our research was the information about how widespread a practice is. As the following quote shows, this actually includes two dimensions:

> *We have an in-house method that covers all system development projects. ... We use it in large projects. There's always a tradeoff, but we use it in the large projects. ... Some large customers, e.g., the military, have strong preferences.* (Senior Developer, Company D)

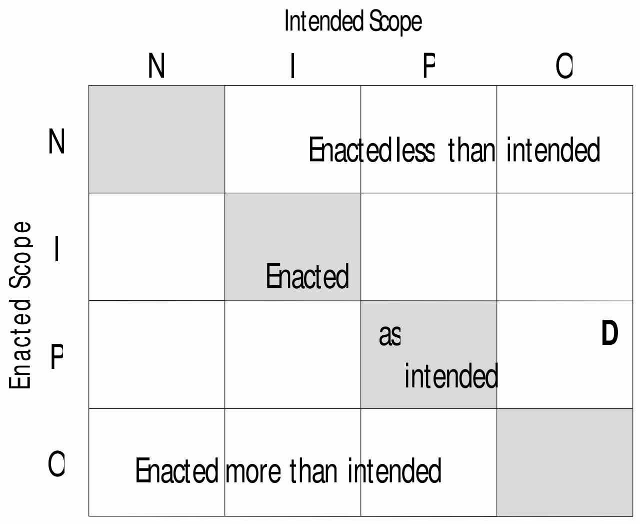

Figure 1: The NIPO grid—intention versus enactment.

The interviewee indicated that his organization's in-house method should (ideally) be used in all ISD projects. This is the *intended scope*—how widespread the practices are intended to be. He also says that the method is used only in large projects. That is, the practices are not enacted as widely as intended. We use the term *actual scope* to denote how widespread the actual use of a practice is. The actual scope can vary from not enacted at all (N) to enacted throughout the organization (O). We found two intermediate levels in our data: Practices could be enacted by some individual(s) (I) or in a project (P). We use the same levels for the intended scope, from a situation where no decision has been made about a practice or a decision has been made not to use it (N) to a situation where a decision has been made to follow the practice throughout the organization (O). The intermediate levels are somewhat arbitrary. Other levels such as department or team are certainly possible. However, the important point here is that the scope may encompass the whole organization or a large or small part of it.

The intended scope and the actual scope of enactment of a practice can be represented in a tabular form in "the NIPO grid" as shown in Figure 1. The method use of Company D, described in the quote above, belongs in the third cell from the top in the rightmost column (cell O,P, marked **D** in the figure).

It is possible to plot the practices found in an organization on the NIPO grid to get visual "snapshots" of its development practices. We can make the following observations about such a plot:

- Practices plotted along the main diagonal of the NIPO grid are enacted exactly as widely as intended. In this ideal case, the developers do what they have decided or have been told to do.[2]
- Practices in the upper-right half of the grid are enacted less widely than intended. Our data contain many instances of this phenomenon. The reason may be as simple as ignorance of the intended practice or may be that the intended practice is not always applicable.
- Practices plotted in the lower-left half are enacted more widely than intended. For example, some practices may be enacted for a time even if official organizational support for them has ceased (an example of this is given in the next section). Other examples are informal patterns of practice that emerge over time.

Appendix 1 includes all identified key practices in the organizations and their location in the NIPO grid.

CHANGE PATHS OF SYSTEMS DEVELOPMENT PRACTICES

We plotted the identified practices on a NIPO grid for each organization. Then we entered a phase of selective coding in which we searched for categories that were linked to the practices and that promoted changes in the scope of practices in the NIPO grid. We found between five and twenty such *forces* in each organization. We plotted the forces on the NIPO grids. An example is shown in Figure 2. However, there were too many differences in the adopted practices and the forces were too contextual to draw any general and meaningful conclusions or even to compare the organizations to each other. Each case had its own history and circumstances that had led to its own set of key practices and their change potential. Analyzing the individual forces of individual practices had been too complex, so we started to analyze how the practices change in relation to the NIPO grid itself.

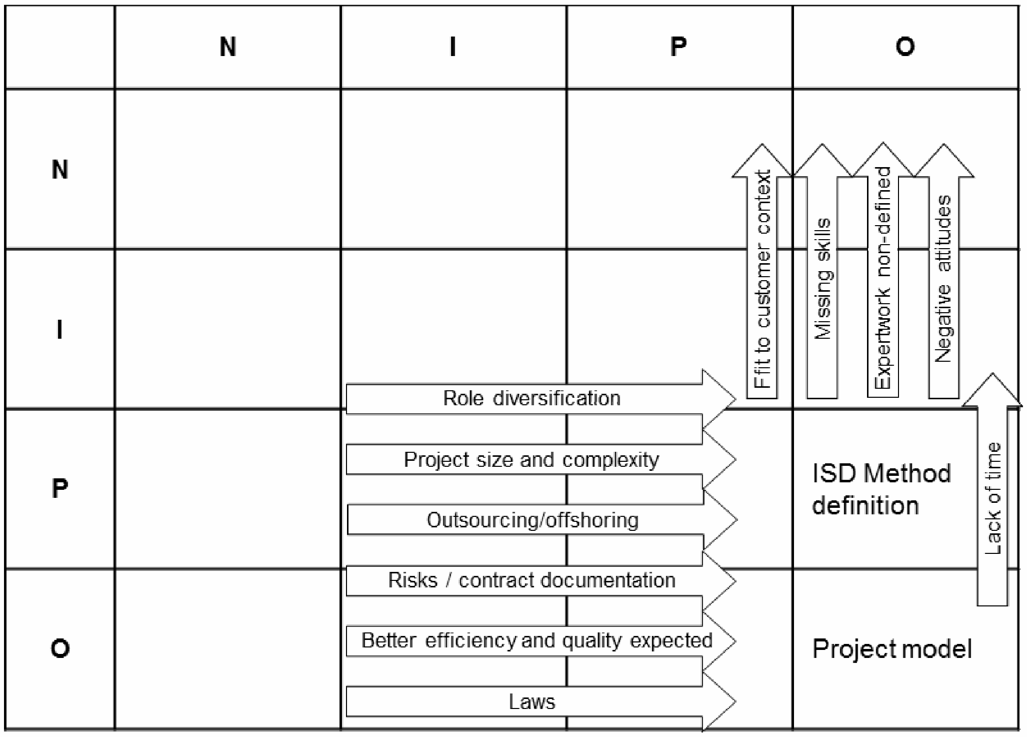

Figure 2: An example of an organization-specific NIPO analysis chart with identified practices and forces influencing their enactment and intended scope.

[2] The cell N,N simply means that a practice does not exist in any form in the organization—it is neither intended nor actually used.

Eight change paths of systems development practices

A practice may change in four elementary directions in the NIPO grid: left, right, up, and down. Leftward and rightward changes correspond to a reduction or increase in the intended scope, that is, a decision to employ a practice more or less widely. An upward or downward change on the other hand corresponds to the change in actual behavior: the number of people enacting the practice is decreasing or increasing. In addition, we can make a distinction between the halves of the NIPO grid. Changes in scope can be made toward or away from the "ideal" main diagonal. This gives us eight basic types of theoretically possible change paths. They are summarized in Table 3. The "Axis" column defines the axis along which the change happens. "From" and "To" describe the state of the practice before and after the change, and the "Example" column gives a typical case of the type of change path in question.

Table 3: Summary of the Eight Stereotypical Change Paths of Practices

Change path	Axis	From	To	Example
Emergence	Actual scope (+)	As intended	More than intended	A practice forms over time through shared experiences.
Entropy	Actual scope (–)	More than intended	As intended	An emergent practice is not followed anymore.
Initiation	Intended scope (+)	As enacted	More than enacted	An organization decides to introduce a practice or to use it more widely.
Abandonment	Intended scope (–)	More than enacted	As enacted	An organization or project decides to abandon a practice that is not enacted.
Formalization	Intended scope (+)	Less than enacted	As enacted	An enacted, emergent practice is formally accepted.
Informalization	Intended scope (–)	As enacted	Less than enacted	A practice is still widely used but no longer "officially" required.
Implementation	Actual scope (+)	Less than intended	As intended	A defined practice is implemented according to a plan.
Recalcitrance	Actual scope (–)	As intended	Less than intended	Developers or projects decide not to follow a defined practice despite its official status.

Empirical evidence of the change paths

This section illustrates how each of the change paths has been grounded on the empirical data gathered from the interviews.

Emergence (Figure 3) happens when a practice becomes enacted more widely than intended. This could happen when, for example, individuals or projects copy successful practices from other individuals or projects. We found several examples of this kind of practice evolution from the data. One example from Company G was already quoted above during the general explanation of the NIPO grid. Another example can be taken from Company D:

> *I am using a spreadsheet to manage the project. I use it to track user stories and burn-down. ... It's just something I made for this project. It's too early to tell how well it works.* (Project Manager, Company D)
>
> *They use the same spreadsheet I saw in the two other projects.* (Field note from ɛ D)

ART: Fix alignment of squares above grid

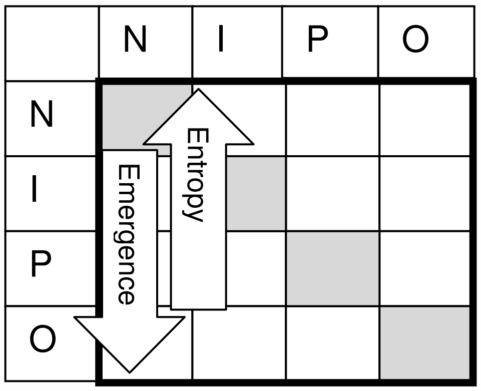

Figure 3: Emergence and entropy.

Entropy (Figure 3) means that an informal practice that was enacted before is no longer or less widely enacted. This may happen, for example, if the practice fits new kinds of tasks badly or if the individuals do not feel confident with it. We could identify examples of a previously emerged practice eroding over time. A developer from Organization B, for example, explained how he had stopped using modeling techniques which he had been using before:

> *Q: You mentioned that you don't have any particular way to do things [design, coding], but you just do your work as you always have done it. Why?*
>
> *A: Now it is just because it works … when I was a freshman I tried to write down stuff beforehand with models and drawings and so on … but now I feel it is an unnecessary step … but if I see a need, I would do it… .* (Developer, Company B)

Initiation (Figure 4) happens when an organization makes a decision to introduce or increase the scope of a practice. Typically the decision is made by a manager or other stakeholder with the power to suggest changes. The practice may be part of a development method, or the management may consider it worthy of broader use after learning about its use by an individual or project. The practice is then explicitly defined and possibly documented, but not yet fully adopted, which requires the process of implementation.

In the data, typical reasons for initiation included expectations of better quality, complexity management, and reduction of risks through better practices. An example is Organization A, which was going to initiate a testing practice by the time of data collection:

> *Yes, there is one area in which we have thought to introduce methods, which is testing and approval. Here we plan to formalize[3] something. A third person can go in to do a quality check and to test. Sometimes the quality produced by developers may vary a bit.* (Senior Developer, Case A)

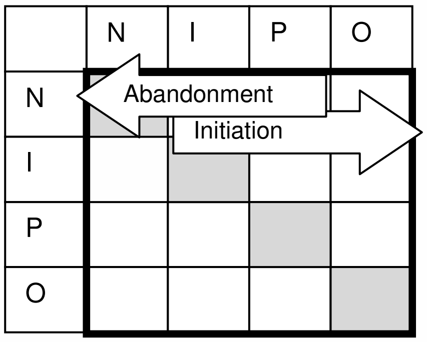

Figure 4: Initiation and abandonment.

Abandonment (Figure 4) means that an organization decides to accept the fact that a defined practice is no longer enacted either in a particular development context or with regard to the whole organization, which is, in a sense, the opposite of Initiation. Possible reasons for this may be that the practice does not work or that it does not give enough benefits in practice. From the data we could find several examples of opinions that the practices do not work or that there are not enough perceived benefits of practices. This can lead to a situation where a decision is made to abandon a practice. The following excerpt from an interview of a Case H developer exemplifies this:

> *Q: Do you have guidelines or common practices related to systems development in your company?*
>
> *A: We have a software development process in our company. This process is ISO certified. We have a number of documents which we follow. The project should have a special format, documents, and discretion in the process about what stages the project should have.*
>
> *Q: How strictly do you follow these documents?*
>
> *A: We are trying to follow them but how strictly, as I said, depends on the customer because always following documents requires additional time for management, and in some cases we do not have this time. So which project documentation will be used for the particular project is agreed on with the customer.*

[3] Note that although the interviewee used the term *to formalize* here, the company was going to initiate a new practice for testing, not formalize an existing one, as is the case below.

Formalization (Figure 5) means that an organization decides to formally accept a widely enacted practice. The practice may have spread through the process of emergence, and formalization is the acceptance of the fact, which includes an explicit definition of the practice. For example, in Case I requirement specification practices had emerged over time without explicit practice definitions. Both customer pressures and the company's own organizational development required it to start formalization of the already existing requirement specification practices:

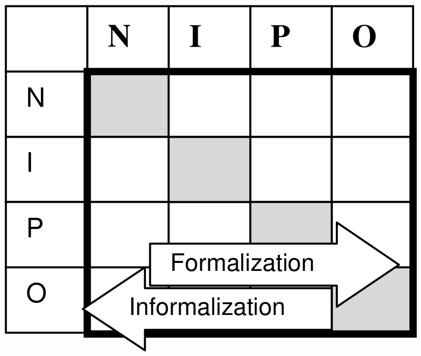

Figure 5: Formalization and informalization.

> *They wish us to define how they can participate in the process. For example, the developers at the customer are doing many tasks for us. Probably half of our designs come directly from the customer—they do it that way—and they would very much like us to show more clearly how they should do it, because they are not software engineers.* (Upper Manager, Case I)

Informalization (Figure 5) happens when the organization formally stops supporting a practice that has previously been widely enacted and formalized. This may happen because the practice in question is not seen as important anymore or because an organizational change removes the sponsors of the practice. An example of the latter is Case C, a former IBM subsidiary that continued its practices in spite of lack of management attention:

> *There is no pressure to follow the guidelines anymore. It was different in IBM.* (Developer, Case C)

The company later went through a formalization process which reintroduced the practices as official policy:

> *We have written a quality manual that is based on the practices carried over from the IBM time.* (Project Manager, Case C)

Implementation (Figure 6) represents a typical idea of how systems development innovations diffuse in organizations (Mustonen-Ollila and Lyytinen 2003) where the actual scope of a practice is increasing after a decision to adopt it has been made. An obvious factor promoting this type of change is the loyalty and obedience of the employees.

In Organization D all projects used a specific set of development practices, "Scrum" (Schwaber and Beedle 2001), because it was the company policy. One developer said he was more likely to follow defined practices if the management strictly coordinated adherence to them.

In Company F the enterprise defined the methods, but they were in a constant process of implementation:

> *Our company offers methods, but we had no experience in those methods and how specifications should be made, what would be the result, and what should be the quality level of the documentation. Therefore, even when we made the specifications for the whole of last year, eventually we were in a hurry and we did not succeed … we can now say that the specifications were not at a sufficiently detailed level.* (Department Manager, Case F)

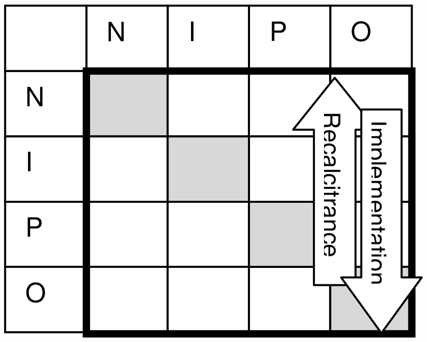

Figure 6: Implementation and recalcitrance.

Recalcitrance (Figure 6) happens when an actor or a project ceases to follow a previously implemented practice. Possible reasons, in addition to those mentioned above, include opposition to the management and lack of training of new employees. For example, some recalcitrance can be observed from the following comment from Case F:

> *This has been a rather crude experience as well. … We did the specification with [the defined method], but I have not familiarized myself with that more, but I guess it is some kind of process or a model of how the specification is executed and what kind of documents are produced in the specification. People tried to learn that and went to some kind of courses. Somehow I got the picture that it did not guarantee anything and it would have forced us to do this and that kind of task.* (Manager, Case F)

In Case I the quality that the practices produced was considered too high and expensive in some projects and, therefore, some individual employees decided not to use the defined practices.

> *I would say that we expect too high quality, because testing and developing currently take quite a time. … I think our quality is probably too good … for our customers' needs; we are just … how to say it … we take too much pride in it working, perhaps.* (Manager, Case I)

DISCUSSION

We have suggested a model of eight change paths through which to analyze the current state-of-the-art and change pressures of systems development practices in context. The model allows systems development practices to be located with the dimensions of intention and enactment and helps analyze and understand the possible change paths of practices. In this section, we discuss the contributions of our results. First, we outline some general-level observations of our model for analyzing developments in systems development practices. Second, we discuss how the model can unify our understanding of previously contradictory viewpoints in the study of the adoption of SDMs versus emerging systems development practices. Third, we point out how this model complements previous research on systems development practices, suggesting a few avenues to guide future research efforts in the field. Finally, we outline some reflections on and shortcomings of our research process, leaving room for additional work.

General observations

In the light of our model, we can make the following propositions that are also supported by the empirical data:

> *The intended scope of definition and the actual scope of enactment of a local development practice can differ.*

For instance, in the example of Company F given above, the practices were defined at the level of the organization, whereas the individual projects often deviated from the defined guidelines. A decision to initiate a practice does not necessarily lead to its enactment. This distinction becomes explicit in light of our model and also shows the difference between the dimensions of learning (actual scope) and decision (intended scope).

> *Different practices may have varying scopes of enactment and definition within and across development organizations.*

Although this statement is also justified by common sense, our analyses of the cases in light of the model (Appendix 1) clearly illustrate these variations. For example, the coding/programming practices in Organizations A and B followed no specific form, but had stricter guidelines and recommendations for contracting and system specifications, respectively. That is, a development organization may introduce strict practices regarding some development tasks while allowing greater individualism and flexibility in others.

Of course, such variance may also exist across organizations. For example, in Company F development practices were defined at the organization level (while varyingly followed), whereas the developers in Companies A and B were rather free to decide many of these individually. Comparisons between organizations may be meaningful with regard to certain types of practices, if topical and popular within the industry. That is, even if the company generally reported "using" a particular set of predefined practices, for example "Scrum," their enactment may vary project-by-project or among individual developers (cf. also Kautz et al. 2004). This observation strengthens the critique and skepticism (e.g., Wynekoop and Russo 1997) of the validity of general-level surveys of development methods and project performance (such as Jenkins et al. 1984; Cusumano and Kemerer 1990).

In the analysis of changes in systems development practices, it is important to distinguish between definition and enactment of practices and between emergent and planned practices.

As an analytical tool, the change path model makes two distinctions which need to be taken into account when researching systems development practices. First, it distinguishes between a definition of a practice and its enactment. The order of these dimensions may vary, practice-by-practice, if practice evolution is analyzed in context. Enactment of a practice (emergence) may precede its definition (formalization), or a practice can be defined first (initiation) and then enacted (implementation). Second, the model illustrates a distinction between emergent and planned practices. The identified change paths show that a practice may become widely enacted and intended through two paths: initiation–implementation or emergence–formalization.

Unifying model for research on ISD method adoption and practice emergence

In previous research, such concepts as "amethodical systems development" (Truex et al. 2000) and "method-in-action" (Fitzgerald et al. 2002) have been presented as alternative, even conflicting, viewpoints to question and oppose the traditional role of prescribed methods in systems development. However, the change path model gives us a basis to cast an integrative look at the previous literature of systems development practice. First, the change paths make it possible to study both preplanned method adoption, as well as emergence of the practices within a joint model. Second, the model highlights how adoption and emergence of development practices in an organizational context may, or perhaps even should, be analyzed simultaneously in relation to multiple units of analyses, as exemplified by the NIPO grid.

To illustrate the first point, let us sketch the concepts of method-in-action (Fitzgerald et al. 2002), amethodical systems development (Truex et al. 2000), and CMM-based software process improvement (Humphrey 1989). We infer that the method-in-action and amethodical systems development concepts represent the research stream studying practice emergence. Respectively, the idea of CMM-based software process improvement (Humphrey 1989) represents the "planned diffusion" stream of adopting ISD methods and practices, in the light of the eight change paths.

In the method-in-action framework (Fitzgerald et al. 2002), a method-in-action refers to the actual practice(s) enacted in a development project. The method-in-action may be based on a formalized method, although this is not obligatory. It is uniquely enacted in each project by the developers. If we break down the method-in-action to the level of practices it can be seen as the collection of all manifest practices that have standardized enactment in the project (which include those possibly standardized and followed throughout the organization). To analyze methods-in-action, our model specifies four possible change paths for identifying how particular methods-in-action may come to be enacted in a development context or project:

- First, a method may emerge tacitly, when developers implicitly or even "instinctively" start to work in commonly adopted routine-like ways through mutual learning and interaction. If such enacted practice begins to be more explicitly identified within a project or development group, it has followed the emergence path from N,N to N,P.
- Second, if an individual is following certain practices which are later imitated by other people in a project who regard such practices as a good idea, the identified method-in-action has emerged from I,I to I,P.
- Third, we also might have cases where a project simply innovates and defines project-specific practices explicitly and collectively from scratch. In that case, the method-in-action does not necessarily "change" in relation to any previous practice or other unit of analysis, and the "emergence" or "practice definition" takes place solely in the project context (P,P).
- Fourth, if an organization has a "formalized method" as its organizational standard, whereas a project alters that method to project-specific practices that deviate from it, the "method-in-action" has resulted from recalcitrance, emerging from O,O to O,P.

Anyhow, if we now want to understand the reasons why some project-specific methods-in-action have been followed, for example, to specify lessons learned from the practice description for future projects, the recognition of the alternative change paths can give a basis for alternative explanations of such histories. In addition, if a researcher may notice that "methods-in-action" might vary even within a single project over time (e.g., if individual developers work in varying ways for similar and comparable purposes), we actually realize that we need to add a unit of analysis to the individual level and search further for explanations for such changes from the project standard. Our model makes explicit these potential choices of the analytical units and dimensions in order to study systems development practices in context within a common frame of reference (Table 4).

Truex et al. (2000) introduced the idea of amethodical system development. In the amethodical view, systems development is an opportunistic process which is unique rather than replicable. A developer's actions will be guided by such factors as mood and social context, rather than logic, plan, and method. Taken to the extreme, amethodical development actions are completely situational. In other words, enacted practices would not be standardized, and seemingly similar practices appear only coincidentally. Consequently, any decision dimension to employ defined practices can be regarded as irrelevant if an analyst studies "amethodical practices" in context. In the light of our model, we infer that identification of such amethodical practices is always explained through contextual emergence. Depending on the context and unit of analysis we might now define whether an individual developer is "amethodical" (i.e., his or her practices vary over time, even within a particular project or development task over time), whether a project is "amethodical" (where we might, in addition, look at variations between individual developers within similar development tasks), or whether a whole development organization is "amethodical," while still having some recognizable patterns of practices. However, if we instead analyze how development work has started to vary in relation to some predefined organizational or project-specific practice, such recalcitrance would not perhaps be "purely amethodical."

Table 4: Methods-in-action, Amethodical Practices, and SPI-related Practices

Actual scope of enacted practice		Intended scope of practice				
		None	**Individual**	**Project**	**Organization**	
			← Abandonment	**–**	**Initiation →**	
None	**← Emergence – Entropy →**	No observable patterns, ad-hoc practices		Initiated SPI-related new practices which are to be implemented for a selected project	Initiated SPI-related new practices which are to be implemented for the whole organization	**← Recalcitrance – Implementation →**
Individual		Individual, emergent amethodical practices				
Project		Emergent method-in-action	Emergent method-in-action	Method-in-action deliberately specified and implemented within a project	Method-in-action deviating (recalcitrance) from an organizational standard	
		Project-specific, emergent amethodical practices	Project-level amethodical practices emerging from individual practices	New SPI-related practices implemented in a selected project or projects (e.g., for piloting a new "best practice")	Project-level recalcitrance against (deviation from) pre-planned practices to be analyzed for SPI	
Organization		Emergent, amethodical organizational practices	Organization-level amethodical practices emerging from individual practices		Implemented practices for SPI which developers accept they will follow in the projects	
			← Informalization	**–**	**Formalization →**	

Software process improvement (SPI) frameworks, such as the Capability Maturity Model (CMM; Paulk et al. 1991), provide yet another perspective on analyzing how development practices change and evolve. The SPI frameworks are designed to assess whether an organization is capable of repeating or even improving its performance from project to project. SPI builds on the assumption that certain predefined practices are desirable. An assessment of an organization is carried out by checking that the organization has defined certain key practices and that they are enacted according to the definition. An SPI assessment will focus on the recommended practices and their documentation. An SPI model can also include “meta-practices” for conscious deviation from the organization-level practice in selected projects in order to try out new things and to improve the hitherto defined process. Used as a research tool this viewpoint assumes that the change paths of initiation and implementation exist within recognized organizational structures that try to find the best ways to plan for and share “best practices.” Eventually, an SPI-inspired researcher may also focus on reasons and explanations for recalcitrance.

The above examples, representing three research streams, indicate considerable variation in both definition and enactment of practices as well as assumed paths for practice evolution and change in systems development projects and organizations. None of the three frameworks alone captures the total dynamic of changing systems development practices in organizations but leaves important scopes of practices and their changes without consideration. This is explicitly shown by the NIPO chart in Table 4, which lists only four of eight possible change types (initiation, emergence, implementation, and recalcitrance).

In addition to these examples, the adoption of SDMs among individual professionals has been a focus of a few empirical studies (e.g., Stolterman 1992; Fitzgerald 1997). The model of intended scope of defined practices versus the actual scope of enacted practices now provides us with an integrated possibility to study how adoption of practices may vary in organizations among individuals and projects. That is, the model highlights how there could be several meaningful units of analysis, simultaneously, when contextual utilization and emergence of practices are to be analyzed.

CMM and other SPI frameworks can be used to analyze systems development practices on a level of detail comparable with the NIPO construct. They also share the organizational focus, but differ by being designed as normative guidelines for improving practice. They build on the assumption that a certain set of key (or “best”) practices should first be defined and then enacted according to the definition. It is outside the scope of an SPI assessment to detect emerging undefined practices. Individual practices and variations may be detected, but will be described as deviations.

Complementing previous research on systems development practice emergence and adoption

In addition to integrating previous research and concepts focusing on ISD method adoption and practice emergence, our model also contributes by suggesting additional change path types according to which evolution and change in practices may be analyzed.

For example, the theory of diffusion of innovation (DoI; Rogers 1995) explains the adoption of ISD process innovations through two phases: the initiation decision and implementation. Mustonen-Ollila and Lyytinen (2003) found that DoI cannot explain all ISD process innovations. The model of eight change paths may shed additional light on this issue. In addition to the initiation decision and implementation, our model complements the DoI theory with six additional types of change paths: emergence, formalization, recalcitrance, abandonment, informalization, and entropy. These include change paths which describe how some innovations come to their end and are abandoned and how new ones may emerge from individual initiatives even without much planning or explicit decisions. In this sense, we argue that the eight identified change paths involve more explanatory power, forming a potential basis for more detailed theoretical elaborations about how systems development practices change in organizations. Also understanding the other potential change paths, in addition to initiation and implementation, may be important, for example, for focused studies on particular idealized practices across multiple projects and organizations. In conclusion, we see that because the DoI theory cannot capture all possible change paths of a practice, a researcher using DoI only to analyze the diffusion of systems development practices may miss important trajectories. For this purpose the eight change paths in the NIPO grid may serve as an important tool for research.

Implications for practice and research

Above we listed some of the strengths of the change path model and also showed how it can be applied to integrating and complementing existing models and theories. Altogether, our work suggests a few implications for practice and research.

For industrial practice, the model can help to understand the enactment of and status of systems development practices in an organization. This knowledge will then enable the organization to focus on those areas of practice where the difference between intentions and actual enactment is the greatest. An “ideal plot” for each domain of development practices or for particular practices would be on the diagonal—that is, where the intentions and enactment would maximally match each other in the organization.

An important feature of the change path model is that it makes a clear distinction between learning a practice and the decision to use the practice. In addition, it does not require that these steps occur in any specific order. We believe that this enables practitioners to better evaluate the status and to plan required actions without overly constraining the innovative nature of organizational development.

For researchers of systems development practice, the model can serve as an analytical tool for building detailed research hypotheses when studying particular types or domains of practices and their evolution in organizational contexts. The two dimensions take into account the granularity of analysis, and the distinction between intentions and enactment may guide a researcher to look deeper into the reality, at whether a practice is really enacted or not, regardless of the decisions made. The model helps to capture variance in practice enactment and intentions which we regard as useful for creating more detailed theories of their rationale and impact. That is, researchers interested in factors affecting implementation of particular idealized practices (such as certain agile practices, requirements engineering practices, testing practices, etc.) can now be more specific on which factors seem to have impact on intentions, whereas other factors may still have impact on their enactment. As well, the model provides a conceptual basis to observe and to form theories of how variance in intentions and enactment of particular practices would impact development performance and resulting systems. Hence, our work contributes to the continuing calls for empirical research and theorizing about systems and software development practice to increase professionalism in the field (e.g., Wynekoop and Russo 1997; Kitchenham et al. 2007). Another use of the model for researchers is to use it as an analytical tool for understanding and integrating hitherto scattered theorizing about systems development practices and change in them. This was illustrated above in relation to a few recent concepts and theories through which development practice has been explained recently.

Methodological reflections

As explained earlier we decided to use Grounded Theory because we wanted to do exploratory research on system development practice, without influences and assumptions from existing theory. During both open and axial coding, we found this ideal of starting with blank sheets to be unachievable. When analyzing the data we would sometimes make use of our earlier experience and theoretical knowledge. We were in doubt as to whether we should try harder to shut out these influences. However Strauss (1987, p. 63) wrote:

> *The analyst does not remain totally bound within the domain of these data, but quickly jumps off to wonder or speculate or hypothesize about data ... . He does this by using his technical knowledge and theoretical sensitivity, his experiential knowledge, and his research knowledge ... . Researchers who are inexperienced in how initial open coding can spring quickly off the data—while yet firmly rooted in it—tend to keep their analyses too limited in scope.*

This is also in line with reflections by Hansen and Kautz (2005) who recognize, based on Walsham (1995), that previous knowledge may not hinder observation of new issues while the researchers keep their minds open. While anchored in our data gathered from the field, our model integrates both previous theoretical ideas, such as the individual, project, and organizational levels of analysis, and the intention versus enactment ideas, while still also providing a new theoretical development in the form of the eight change paths in the light of the integrated ideas.

Another potential source of bias is in the data itself. The interviews reflect the respondents' rationalized views of their own practices, which are not necessarily the same as the actual practices. This rationalizing can hide variations in how a practice is enacted and can make it difficult to determine how widespread the practice actually is. Ideally the interviews should be combined with observation of practices being carried out. Field observation, however, is prohibitively expensive in terms of man-hours of both researchers and practitioners, and, therefore, it is not always acceptable to the practitioner organizations or economical for researchers.

One of the challenging issues in creation of a grounded theory is the question of theoretical saturation (Strauss and Corbin 1990). It is often difficult to determine whether more data would add new important features to the created theory. Although we cannot be absolutely sure about the theoretical saturation of our data collection, we found evidence of practices along all columns and rows in the grid, and all directions in which a practice can change on the NIPO grid are covered by examples. There are obviously many possibilities for more fine-grained analysis and richer variation of concepts which might be selected to describe a practice and its particular types of change paths in more detail that do not come up in our observations. Still, we believe that the study shows enough saturation for the identification of the eight types of change paths at an abstract level, in the light of the two dimensions of intended scope of defined practices versus the actual scope of enacted practices. Other organizational units of analysis could be identified instead of or in addition to the NIPO grid levels of individuals and projects, such as teams, professional roles, and product lines. Still, we argue that the NIPO grid serves in this paper as a satisfactory means for illustrating the idea of the change paths, which represents the main theoretical contribution of our work.

Our initial aim to build up theories about *why* changes happen in particular types of systems and software development practices is still to be fulfilled. However, we consider that our model provides a promising basis for further theorizing on how and why systems development practices change. Longitudinal follow-ups of change in particular practices, explaining reasons and more detailed paths for change in the light of our model, remain on our future research agenda. Another area of future research resides in the question of whether and how changes in particular practices would impact development performance and results.

CONCLUSIONS

We have presented the results of an empirical study on systems development practices in eleven systems development organizations. The first result of the study is a model that describes how the intended scope of practice definition and the actual scope of practice enactment either match or differ in organizations. In light of this model, we studied further the issue of how development practices in systems and software development organizations are shaped and changed. This resulted in identification of eight stereotypical change paths through which a particular practice may evolve: emergence, entropy, initiation, abandonment, formalization, informalization, implementation, and recalcitrance. Finally, in the discussion we demonstrated how our research integrates and complements previous theories and concepts of systems development practice, such as "method-in-action" (Fitzgerald et al. 2002), "amethodical systems development" (Truex et al. 2000), and CMM-based software process improvement (Humphrey 1989).

We expect that the model will aid in clarifying organizational differences and similarities in adoption of systems development practices, thus functioning as a useful construct in such theorizing. The model provides an explicit vocabulary for explaining practices and their change in systems development organizations. Our future plans include using the construct further for better understanding of the reasons behind variation in development practices between organizations, where it would be meaningful to make such comparisons. The construct can also be used as one basis for explaining the impact of variance in particular practices (within and between organizations) on the outcomes of systems development. The model can guide a researcher to look beyond the surface of local systems development practices to see whether or not a practice is really enacted, regardless of the decisions made. For practitioners, the grid may help understand the status of local practice repertoires. We believe that because the model makes a clear distinction between emergence and implementation, it enables practitioners to better evaluate the status and plan process innovations without overly constraining the innovative nature of organizational development.

REFERENCES

Argyris, C., and D.A. Schön, *Theory in Practice: Increasing Professional Effectiveness*, Oxford: Jossey-Bass, 1974.

Avison, D.E., and G. Fitzgerald, *Information Systems Development: Methodologies, Techniques and Tools* (3rd edition), London: McGraw-Hill, 2003.

Avison, D.E., and A.T. Wood-Harper, "Information systems development research: An exploration of ideas in practice," *The Computer Journal*, 1991, 34:2, pp. 98–112.

Bajec, M., D. Vavpotic, and M. Krisper, "Practice-driven approach for creating project-specific software development methods," *Information and Software Technology*, 2007, 49:4, pp. 345–365.

Bansler, J.P., and K. Bødker, "A reappraisal of structured analysis: Design in an organizational context," *ACM Trans. Information Systems*, 1993, 11:2, pp. 165–193.

Baskerville, R., B. Ramesh, L. Levine, J. Pries-Heje, and S. Slaughter, "Is Internet-speed software development different?" *IEEE Software*, 2003, 20:6, pp. 70–77.

Beck, K., *Extreme Programming Explained: Embrace Change*, Boston, MA: Addison-Wesley, 1999.

Blaha, M.R., W.J. Premerlani, and J.E. Rumbaugh, "Relational database design using an object-oriented methodology," *Communications of the ACM*, 1988, 31:4, pp. 414–427.

Booch, G., *Object-oriented Design with Applications*, Redwood City, CA: Benjamin-Cummings, 1991.

Brown, J.S., and P. Duguid, “Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation,” *Organization Science*, 1991, 2:1, pp. 40–57.

Coad, P., and E. Yourdon, *Object Oriented Analysis* (2nd edition), Upper Saddle River, NJ: Yourdon Press, 1990.

Coad, P., and E. Yourdon, *Object Oriented Design*, Upper Saddle River, NJ: Yourdon Press, 1991.

Constantine, L.L., and E. Yourdon, *Structured Design: Fundamentals of a Discipline of Computer Programme and Systems Design*, Englewood Cliffs, CA: Yourdon Press, 1979.

Cusumano, M.A., and C.F. Kemerer, “A quantitative analysis of U.S. and Japanese practice and performance in software development,” *Management Science*, 1990, 36:11, pp. 1384–1406.

Davis, G.B., “Strategies for information requirements determination,” *IBM Systems Journal*, 1982, 21:2, pp. 4–30.

Denzin, N.K., *The Research Act: A Theoretical Introduction to Sociological Methods*, New York: McGraw-Hill, 1978.

Dijkstra, E., “Programming considered as a human activity,” In Yourdon, E. (ed.), *Classics in Software Engineering*, Upper Saddle River, NJ: Yourdon Press, 1979.

Fitzgerald, B., “The use of systems development methodologies in practice: A field study,” *Information Systems Journal*, 1997, 7:3, pp. 201–212.

Fitzgerald, B., “An empirical investigation into the adoption of systems development methodologies,” *Information & Management*, 1998a, 34:6, pp. 317–328.

Fitzgerald, B., “An empirically-grounded framework for the information systems development process,” *Proceedings of the International Conference on Information Systems (ICIS)*, AIS, 1998b, pp. 103–114.

Fitzgerald, B., G. Hartnett, and K. Conboy, “Customising agile methods to software practices at Intel Shannon,” *European Journal of Information Systems*, 2006, 15:2, pp. 200–213.

Fitzgerald, B., N.L. Russo, and T. O’Kane, “Software development method tailoring at Motorola,” *Communications of the ACM*, 2003, 46:4, pp. 64–70.

Fitzgerald, B., N.L. Russo, and E. Stolterman, *Information Systems Development: Methods in Action*, London: McGraw-Hill, 2002.

Gane, C., and T. Sarson, *Structured Systems Analysis: Tools and Techniques*, Englewood Cliffs, NJ: Prentice-Hall, 1979.

Glaser, B.G., *Theoretical Sensitivity*, Mill Valley, CA: Sociology Press, 1978.

Glaser, B.G., and A.L. Strauss, *The Discovery of Grounded Theory*, New York: Aldine de Gruyter, 1967.

Hansen, B.H., and K. Kautz, “Grounded theory applied—Studying information systems development methodologies in practice,” *Proceedings of the 38th Hawaii International Conference on System Sciences* [CD-ROM], 2005.

Hardgrave, B.C., F.D. Davis, and C.K. Riemenschneider, “Investigating determinants of software developers’ intentions to follow methodologies,” *Journal of Management Information Systems*, 2003, 20:1, pp. 123–151.

Humphrey, W.S., *Managing the Software Process*, Boston, MA: Addison-Wesley Longman Publishing Co., 1989.

Humphrey, W.S., T.R. Snyder, and R.R. Willis, “Software process improvement at Hughes Aircraft,” *IEEE Software*, 1991, 8:4, pp. 11–23.

Iivari, J., “A methodology for IS development as an organisational change: A pragmatic contingency approach,” In Klein, H., and K. Kumar (eds.), *Information Systems Development for Human Progress in Organisations*, Amsterdam: North-Holland, 1989, pp. 197–217.

Iivari, J., R. Hirschheim, and H.K. Klein, “A paradigmatic analysis contrasting information systems development approaches and methodologies,” *Information Systems Research*, 1998, 9:2, pp. 164–193.

Iivari, J., R. Hirschheim, and H.K. Klein, “Beyond methodologies: Keeping up with information systems development approaches through dynamic classification,” *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 1999.

Iivari, J., and K. Lyytinen, “Research on information systems development in Scandinavia—Unity in plurality,” *Scandinavian Journal of Information Systems*, 1998, 10:1, pp. 135–185.

Introna, L., and E. Whitley, “Against method-ism: Exploring the limits of method,” *Information Technology & People*, 1997, 10:1, pp. 31–45.

ISO/IEC, ISO/IEC 15504-1, *Information Technology—Process Assessment—Part 1: Concepts and Vocabulary*, 2002.

ISO/IEC, ISO/IEC 12207:2008 *Systems and Software Engineering—Software Life Cycle Processes*, 2008.

Jackson, M.A., “The general and the particular,” *Challenges and Strategies for Research in Systems Development*, New York: Wiley, 1992, pp. 33–40.

Jacobson, I., *Object-oriented Software Engineering: A Use Case Driven Approach,* Boston, MA: Addison-Wesley, 1992.

Jarke, M., K. Pohl, C. Rolland, and J.R. Schmitt, “Experience-based method evaluation and improvement: A process modelling approach,” *Proceedings of the IFIP WG8. 1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle*, New York: Elsevier Science, pp. 1–27.

Jayaratna, N., *Understanding and Evaluating Methodologies: NIMSAD, a Systematic Framework,* New York: McGraw-Hill, 1994.

Jenkins, A.M., J.D. Naumann, and J.C. Wetherbe, “Empirical investigation of systems development practices and results,” *Information & Management*, 1984, 7, pp. 73–82.

Kautz, K., “The enactment of methodology: The case of developing a multimedia information system” *Proceedings of the International Conference on Information Systems*, Washington, DC, 2004.

Kautz, K., B. Hansen, and D. Jacobsen, “The utilization of information systems development methodologies in practice,” *Journal of Information Technology Cases and Applications*, 2004, 6:4, pp. 1–19.

Kautz, K., S. Madsen, and J. Nørbjerg, “Persistent problems and practices in information systems development,” *Information Systems Journal*, 2007, 17, pp. 217–239.

Khalifa, M., and J.M. Verner, “Drivers for software development method usage,” *IEEE Transactions on Engineering Management*, 2000, 47:3, pp. 360–369.

Kitchenham, B., D. Budgen, P. Brereton, M. Turner, S. Charters, and S. Linkman, “Large-scale software engineering questions—Expert opinion or empirical evidence?” *IET Software*, 2007, 1:5, pp. 161–171.

Kogut, B., and U. Zander, “Knowledge of the firm, combinative capabilities, and the replication of technology,” *Organization Science*, 1992, 3:3, pp. 383–397.

Kostova, T., and K. Roth, “Adoption of an organizational practice by subsidiaries of multinational corporations: Institutional and relational effects,” *Academy of Management Journal*, 2002, 45:1, pp. 215–233.

Kumar, K., and R.J. Welke, “Methodology engineering: A proposal for situation-specific methodology construction,” In Cotterman, W.W., and J.A. Senn (eds.), *Challenges and Strategies for Research in Systems Development*, New York: Wiley, 1992, pp. 257–269.

Locke, K., *Grounded Theory in Management Research*, Thousand Oaks, CA: Sage, 2001.

Lundeberg, M., G. Goldkuhl, and A. Nilsson, *Information System Development: A Systematic Approach,* Englewood Cliffs, CA: Prentice Hall, 1981.

Lyytinen, K., *Information Systems Development as Social Action: Framework and Critical Implications,* Ph.D Dissertation, Jyväskylän Studies in Computer Science, 1986.

Lyytinen, K., “A taxonomic perspective of information systems development: Theoretical constructs and recommendations,” *Critical Issues in Information Systems Research*, New York: John Wiley & Sons, 1987, pp. 3–41.

Madsen, S., K. Kautz, and R. Vidgen, “A framework for understanding how a unique and local IS development method emerges in practice,” *European Journal of Information Systems*, 2006, 15:2, pp. 225–238.

Martin, J., and C. Finkelstein, *Information Engineering*, vol 1 & 2, Englewood Cliffs, NJ: Prentice-Hall, 1981.

Martin, P.Y., and B.A. Turner, “Grounded theory and organizational research,” *The Journal of Applied Behavioral Science*, 1986, 22:2, p. 141.

Mathiassen, L., “Reflective systems development,” *Scandinavian Journal of Information Systems*, 1998, 10:1&2, pp. 67–118.

Mathiassen, L., and S. Purao, “Educating reflective systems developers,” *Information Systems Journal*, 2002, 12:2, pp. 81–102.

Mathiassen, L., and L. Vogelsang, “The role of networks and networking in bringing software methods to practice,” *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005, pp. 256b–256b.

Mustonen-Ollila, E., and K. Lyytinen, “Why organizations adopt information system process innovations: A longitudinal study using Diffusion of Innovation Theory,” *Information Systems Journal*, 2003, 13:3, pp. 275–297.

Nelson, R.R., and S.G. Winter, *An Evolutionary Theory of Economic Change*, Cambridge, MA: Belknap Press, 1982.

Nilsson, T., *Smidig utvikling i en mellomstor bedrift på Sørlandet* (in Norwegian), Master’s Thesis, Kristiansand, Norway: Department of Information Systems, University of Agder, 2008.

Orlikowski, W.J., “Knowing in practice: Enacting a collective capability in distributed organizing,” *Organization Science*, 2002, 13:3, pp. 249–273.

Päivärinta, T., E.Å. Larsen, L. Ingulfsvann, and Ø. Kristiansen, “Method deployment in small- and medium-sized firms: Addressing the organizational context,” In Wojtkowski, W., G. Wojtkowski, M. Lang, K. Conboy, and C. Barry (eds.), *Information Systems Development: Challenges in Practice, Theory, and Education*, Vol. 1, New York: Springer, 2009, pp. 65–77.

Päivärinta, T., M.K. Sein, and T. Peltola, “From ideals towards practice: Paradigmatic mismatches and drifts in method deployment,” *Information Systems Journal*, 2010, 20:5, pp. 481–516.

Parnas, D.L., “On the criteria to be used in decomposing systems into modules,” *Communications of the ACM*, 1972, 15:12, pp. 1053–1058.

Paulk, M.C., B. Curtis, M.B. Chrissis, and C.W. Weber, *Capability Maturity Model for Software*, Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, 1991.

Pentland, B.T., and M.S. Feldman, “Organizational routines as a unit of analysis,” *Industrial and Corporate Change*, 2005, 14:5, pp. 793–815.

Reichertz, J., “Abduction: The logic of discovery of grounded theory,” In Bryant, A., and K. Charmaz (eds.), *The Handbook of Grounded Theory*, London: Sage, 2007, pp. 339–360.

Rogers, E.M., *Diffusion of Innovations* (4th edition), New York: The Free Press, 1995.

Rowlands, B., “The enactment of methodology: An institutional account of systems developers as social actors,” *Scandinavian Journal of Information Systems*, 2008, 20:2, pp. 21–48.

Royce, W.W., “Managing the development of large software systems,” *Proceedings of IEEE Wescon* (Vol. 26, p. 9), 1970, pp. 328–338.

Schwaber, K., and M. Beedle, *Agile Software Development with Scrum*, Upper Saddle River, NJ: Prentice Hall, 2001.

Siau, K., and M. Rossi, “Evaluation techniques for systems analysis and design modelling methods—A review and comparative analysis,” *Information Systems Journal*, 2011, 21:3, pp. 249–268.

Software Engineering Institute, *CMMI for Development*, Version 1.2., 2006.

Stevens, W.P., G.J. Myers, and L. L. Constantine, “Structured Design,” *IBM Systems Journal*, 1974, 13:2.

Stolterman, E., “How system designers think about design and methods: Some reflections based on an interview study,” *Scandinavian Journal of Information Systems*, 1992, 4:1, pp. 137–150.

Strauss, A.L., *Qualitative Analysis for Social Scientists*, Cambridge, MA: Cambridge University Press, 1987.

Strauss, A.L., and J.M. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory,* Thousand Oaks, CA: Sage Publications, 1990.

Szulanski, G., “Exploring internal stickiness: Impediments to the transfer of best practice within the firm,” *Strategic Management Journal*, 1996, 17:Winter Special Issue, pp. 27–43.

Tolvanen, J.P., *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence,* Jyväskylä Studies in Computer Science, Economics and Statistics, University of Jyväskylä, 1998.

Truex, D., R. Baskerville, and J. Travis, “Amethodical systems development: The deferred meaning of systems development methods,” *Accounting, Management and Information Technologies*, 2000, 10:1, pp. 53–79.

Urquhart, C., “The evolving nature of grounded theory method: The case of the information systems discipline,” In Bryant, A., and K. Charmaz (eds.), *The Handbook of Grounded Theory*, London: Sage, 2007, pp. 339–360.

Vidgen, R., and X. Wang, “Coevolving systems and the organization of agile software development,” *Information Systems Research*, 2009, 20:3, pp. 355–376.

Vogelsang, L., and F. Kensing, “Utilizing systems development methods—A conceptual framework,” *Proceedings of the 14th European Conference on Information Systems*, Göteborg, Sweden, 2006.

Walsham, G., “Interpretive case studies in IS research: Nature and method,” *European Journal of Information Systems*, 1995, 4:2, pp. 74–81.

Whitley, E.A., “Method-ism in practice: Investigating the relationship between method and understanding in Web page design,” *Proceedings of the International Conference on Information Systems*, Helsinki, Finland, Atlanta, GA: Association for Information Systems, 1998.

Wirfs-Brock, R., and B. Wilkerson, “Object-oriented design: A responsibility-driven approach,” *Conference Proceedings on Object-oriented Programming Systems, Languages, and Applications*, New Orleans, LA, 1989.

Wynekoop, J.L., and N. Russo, “Studying system development methodologies: An examination of research methods,” *Information Systems Journal*, 1997, 7, pp. 47–65.

Yourdon, E., *Modern Structured Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

APPENDIX 1: KEY PRACTICES AND THEIR NIPO LOCATION

Practice	Case	NIPO Location
Programming	A	(N/I)-I
Testing	A	(N/I)-I
Architecture	A	O-P
Code reuse	A	O-P
Practice of “no support”	A	O-O
Project “management”	A	O-O
Contract negotiation	A	O-O
Improvised documentation practice	B	I-I
Documented development process	B	O-N
Iterative UI development	B	O-O
Product and database specification	B	O-O
Testing specification	B	O-O
Specification practices in general	B	O-O
Quality system	C	O-P
Programming/Design/Testing	D	N-I
Project backlog administration	D	P-I
Sprint daily meeting	D	P-P
Sprint length	D	P-P
Project administration	D	P-O
Effort estimation	D	O-O
In-house method	E	O-P
ISD method definition	F	O-I/P
Project model	F	O-O
Change management	G	N-O
Project model	G	N-O
Functional design	G	I/P-O
Coding	G	O-O
Installation	G	O-O
Sales	G	O-O
Task list/Work planning	G	O-O
Process model	H	O-P
Effort estimation	H	O-O
Coding style	H	O-O
Configuration management	H	O-O
Requirement specification	I	N-P
Design	I	P-P
Task list	I	O-O
Change, testing, documentation, and coding processes	I	O-O
Project model, including deployment, documentation, and change management	J	O-O
Software acquisition	J	O-O
Information/Database modelling (IDEF1X)	K	P-I
Data-flow-diagramming for analysis (Excelerator, Systems Engineer)	K	P-N
Change management for the database model	K	P-I
Selling development hours for internal customers (customer-driven development)	K	O-O
Ordering/Change process for developing new functionality (ITIL)	K	O-O
Program code change management, documenting change specifications	K	N-I
Scrum: Sprint	K	P-I
Scrum: Product backlog	K	N/I-P
Test cases for testing	K	N-P

APPENDIX 2: KEY PRACTICES, CHANGE PRESSURES AND PRESSURE DIRECTIONS

The table below includes only those practices (cf. Appendix 1) identified with “pressures” to change.

Practice	Case	Pressure to change	Change path direction
Testing	A	Quality improvement	Initiation/Formalization
Practice of “no support”	A	No profits available	Formalization
Improvised documentation practice	B	Reduce risks	Emergence
Documented development process	B	Efficiency	Abandonment
Iterative UI development	B	Manager competence	Informalization
Product and database specification	B	Culture/Competence	Informalization/Formalization
Testing specification	B	Physical distance	Formalization
Specification practices in general	B	Ignorance	Recalcitrance
Quality system	C	Staff flexibility	Formalization
	C	Quality enhancement	Implementation
	C	Standardization	Implementation
	C	Attitudes	Recalcitrance
Programming/Design/Testing	D	Quality improvement	Formalization
Project backlog administration	D	Unclear motivation	Recalcitrance
	D	Customers lacking competence	Abandonment
Sprint daily meeting	D	Customer commitment	Abandonment
	D	Team locations	Abandonment
Sprint length	D	Customer feedback	Abandonment
Project administration	D	Success experiences	Implementation
Effort estimation	D	Success experiences	Implementation
In-house method	E	Success experiences	Formalization
	E	Improved documentation	Formalization
	E	Slow decision making	Recalcitrance
	E	Method customization	Recalcitrance
	E	Customer willing to pay for documentation	Implementation
	E	Project size	Implementation
ISD method definition	F	Negative attitudes	Recalcitrance
	F	Expert work non-definable	Recalcitrance
	F	Missing skills	Recalcitrance
	F	No fit to customer context	Recalcitrance
	F	Role diversification	Formalization
	F	Project size and complexity	Formalization
	F	Outsourcing/Offshoring	Formalization
Project model	F	Lack of time	Recalcitrance
	F	Risks/Contract documentation	Formalization
	F	Better efficiency and quality expected	Formalization
	F	Laws	Formalization
Change management	G	Organizational culture & evolution	Emergence
Project model	G	Organizational culture & evolution	Emergence
Functional design	G	Customer specificity	Entropy
	G	Expert work embedded in persons	Entropy
	G	Too complex practices	Entropy
	G	Lack of time	Entropy
	G	Learning enablement	Formalization
	G	Better quality	Formalization
	G	Personal interests	Formalization
Coding	G	Better documentation	Formalization
	G	Artistry	Recalcitrance
Task list/Work planning	G	Work planning	Formalization
Process model	H	Customer pressures	Formalization
	H	Face value	Formalization
	H	Inter-organizational processes	Formalization
	H	Language and cultural differences	Formalization

Practice	Case	Pressure to change	Change path direction
	H	Customer processes	Recalcitrance
	H	Application domain differences	Recalcitrance
	H	Need for quick results	Recalcitrance
	H	Lack of time	Recalcitrance
	H	High change rate	Recalcitrance
	H	Personal differences	Recalcitrance
Effort estimation	H	Fixed price	Formalization
Requirement specification	I	Tasks done by customer	Entropy
	I	Customer expectations	Formalization
Design	I	Varying project sizes	Recalcitrance
	I	Costs of varying practices	Formalization
Task list	I	Work allocation	Formalization
	I	Activity monitoring	Formalization
	I	Added confidence	Formalization
	I	Small project sizes	Formalization
	I	Varying project sizes	Recalcitrance
Change, testing, documentation, and coding processes	I	Work guidance	Formalization
	I	Influence from professional communities	Formalization
	I	Quality becomes too expensive	Recalcitrance
	I	Documentation is too heavy	Recalcitrance
Project model, including deployment, documentation, and change management	J	Subcontracting/outsourcing	Formalization
	J	Vendor influence	Formalization
	J	Influence from professional communities	Formalization
	J	Standards	Formalization
	J	Different languages and cultures	Formalization
	J	Quality and risk management	Formalization
	J	Project size and complexity	Formalization
	J	Protection from competition	Formalization
	J	Technology differences	Recalcitrance
	J	Varying project sizes	Recalcitrance
	J	Expert work non-definable	Recalcitrance
	J	Customer influence	Recalcitrance
Information/Database modeling (IDEF1X)	K	Benchmarking contemporary (normative) research on SDMs	Initiation/Implementation
Data-flow-diagramming for analysis (Excelerator, Systems Engineer)	K	Benchmarking contemporary (normative) research on SDMs	Initiation/Implementation
	K	Method/tool did not support the implementation phase	Recalcitrance/Abandonment
	K	Model could not keep up with implementation changes	Informalization/Entropy
Change management for the database model	K	Problems to keep track on database model changes done by individual developers	Initiation
Selling development hours for internal customers (customer-driven development)	K	No incentives for documenting & coordinating changes, leads to problems	Recalcitrance/Abandonment
Ordering/Change process for developing new functionality (ITIL)	K	Regarded as “bureaucratic”	Recalcitrance
Program code change management, documenting change specifications	K	Problems to coordinate functionality updates, application quality problems	Initiation/Formalization
	K	Turnover of personnel	Entropy
Scrum: Sprint	K	Requirement changes during a	Recalcitrance

Practice	Case	Pressure to change	Change path direction
		sprint	
Scrum: Product backlog	K	Continuous customer feedback	Implementation
	K	Lack of experience to lead scrum-practices	Recalcitrance
Test cases	K	Ad hoc imitation of previous projects	Emergence
	K	Lack of developer interest to involve in test cases after sprint is done	Formalization

ABOUT THE AUTHORS

Even Åby Larsen is an associate professor in information systems at the University of Agder, Kristiansand, Norway. He has a Master's degree in computer science (1988) from the University of Oslo. He worked with applied research at the Norwegian Computing Center for eight years, and then as a system development consultant and software architect, before joining the university in 2002. His research areas are the information system development process, and system development education.

Tero Päivärinta is professor of computer and systems science at Luleå University of Technology, Sweden, and professor II of information systems at University of Agder, Kristiansand, Norway. He holds a Ph.D. in economics from the University of Jyväskylä, Finland. In addition to systems development practice, his research areas have focused on the methodological issues on using Delphi studies in the field of information systems, benefits realization from IT investments, e-government, and enterprise content management. Tero's works have been previously published in prominent IS journals such as *European Journal of Information Systems*, *Information Systems Journal*, *Scandinavian Journal of Information Systems*, *Information & Software Technology*, *Communications of the AIS*, *Information & Organization*, and in the leading IS conferences.

Kari Smolander is Professor of Software Engineering in the Department of Information Technology, Lappeenranta University of Technology, Finland. He has a Ph.D. (2003) in Computer Science from Lappeenranta University of Technology and a Licentiate (1993) and Master's (1988) degree from University of Jyväskylä, Finland. In addition to his long teaching experience, he has worked several years in industry, being responsible of software product development. His current research interests include architectural aspects of systems development and organizational view of software development.

JITTA

JOURNAL OF INFORMATION TECHNOLOGY THEORY AND APPLICATION

JITTA IS A PUBLICATION OF THE ASSOCIATION FOR INFORMATION SYSTEMS
ISSN: 1532-3416