Scandinavian Journal of Information Systems

Volume 21 | Issue 2 Article 5

12-31-2009

The Relationships Between Competence, Methods, and Practice in Information Systems Development

Hans O. Omland
University of Agder, Hans.O.Omland@uia.no

The relationships between competence, methods, and practice in information systems development

Hans Olav Omland
University of Agder, Kristiansand, Norway
Hans.O.Omland@uia.no

Abstract. This paper investigates the relationships that unfold between an actor's competence, methods, and practice during information systems development (ISD). The data was gathered in a case study of a successful ISD in a Norwegian municipality. In theory, competence, methods, and practice are separate and clearly distinct elements. In actual ISD, however, the three elements form close and integrated relationships. While previous research has addressed some of the relationships between competence, methods, and practice, researchers have yet to describe fully how the three elements relate to and influence each other. This paper's main contribution is a new and more detailed understanding of the tight and intrinsic relationships between competence, methods, and practice and how the three elements dynamically influence each other during ISD processes. The result is a deeper understanding of the ISD process that will help systems developers better establish, monitor, and succeed in their ISD projects.

Key words: Competence, methods, practice, systems development.

1 Introduction

ISD is "an intentional change process which is driven by certain more or less clear objectives" (Mathiassen 1998, p. 70). While actors perform the change process in a context that includes a set of social and technical factors, the change process itself is shaped and influenced by many factors, "including the experiences and competence of the development group" and "the dynamics of the objectives" (Mathiassen 1998, p. 70). Researchers have made many attempts to describe

Accepting editor: Matti Rossi

ISD methods. Fitzgerald et al. (2002) coined the 'method-in-action' concept to denote how a method "is uniquely enacted by the developer" (p. 13). This enactment is shaped by the development context and influenced by the role of methods (Fitzgerald et al. 2002). Madsen et al. (2006) extend this line of research, suggesting a framework that explains how a unique and local method emerges over time in a complex interplay between human action, structural elements, and the 'emergent method'.

Although both Fitzgerald et al. (2002) and Madsen et al. (2006) clearly centre their research around method, the scope of their ISD discussions widen to include both the developer and other contingency factors. This widening of scope recognises that, while systems development might be informed by methods, it depends on much more than methods.

Human actors develop information systems using whatever competence they have in the chosen methods or method elements. Competence and methods, used in practice, are key elements in ISD processes. It is therefore important to understand and describe the relationships between them in actual ISD situations. Some research describes the relationship between methods and practice (e.g., Fitzgerald et al. 2002; Madsen et al. 2006) or competence and methods (e.g., Mathiassen and Purao 2002); there is less existing work on the relationship between competence and practice (e.g., Mathiassen and Purao 2002). Section 2 further reviews the literature on these relationships. No research has been found that describes all three relationships and how the elements influence each other. This research therefore seeks to answer the following question: How do competence, methods, and practice relate to and influence each other in ISD?

This paper reports on a case study of a successful ISD for a Norwegian municipality. The case data and analysis of it form the basis for describing the relationships between these three elements in the ISD project.

The paper is structured as follows: The next section presents an overview of research on competence, methods, and practice and the relationships between them. Section 3 describes the research approach. A case description and case analysis follow in sections 4 and 5, respectively. The relationships between competence, methods, and practice are then discussed in section 6, followed by the conclusion in section 7.

2 Methods, competence, and practice

2.1 The elements

Developers typically devise methods to make the ISD process simpler and more controllable. Fitzgerald et al. (2002) define a method as "a coherent and systematic approach, based on a particular philosophy of systems development, which will guide developers on what steps to take, how these steps should be performed and why these steps are important in the development of an information system" (p. 5). In this paper, methods are understood to cover larger or smaller parts of ISD. Also, the term 'method elements' is sometimes used to describe parts of a method.

Methods differ dramatically and often address different objectives (Avison and Fitzgerald 1995). They are also based on many implicit and explicit assumptions and views (Iivari and Hirschheim 1996). An analysis of 10 Scandinavian ISD approaches shows that concepts such as scope, value orientation, knowledge interest of ISD, the role of methods, and the principle of the ISD process are used (Iivari and Lyytinen 1998, p. 162). Because the focus of this paper is to research the relationships between methods, competence, and practice in ISD, it is important to investigate how developers view and use methods. Ørvik et al. (1999) describe four versions of the same method depending on how it is understood and deployed. The first version formally describes the method, while the other versions relate more to its actual deployment—that is, how the developer interprets and understands the method, how the organization as a whole adopts it, and how it is actually enacted in an ISD process. ISD actors might benefit more from "tools that help to identify and process the emerging conflicts than tools that aid in developing a technically 'perfect' and optimized solution" (Smolander and Rossi (2008), p. 37).

Competence is deployed by human actors in ISD processes. The competence concept is used in many ways and in different areas of research (Bassellier et al. 2001), and many different conceptualisations are suggested. Still, researchers seem to agree on a generic conceptualization of an individual's competence as a combination of three elements: cognitive competence, skills, and affective competence (Marcolin et al. 2000 referring Kraiger et al. 1993). These three categories entail three important abilities: cognition (the ability to think); skills (the ability to do something); and affections (the ability to relate to other people). Lee et al. (1995) define four broad categories of critical skills and knowledge requirements for IS professionals: (a) technical, (b) technology management, (c) business functional, and (d) interpersonal and management knowledge/skills. Categories (a) – (c) relate to both cognitive competence and skills, while (d) relates to affective competence. White and Leifer (1986) suggest five competencies that contribute to successful systems development: business knowledge, good communication skills, technical expertise, analytical skills, and good organizational skills. These competencies group along the generic conceptualisation of Kraiger et al. (1993).

Competence is the ability, or enabler, that provides the means for performance (Bassellier et al. 2001). Using competence and performance interchangeably will lead to confusion (Bassellier et al. 2001). Referring to Schaumbach (1994), they state that the terms are related, but that "factors other than competence—such as motivation, effort, and supporting conditions—may influence performance" (Bassellier et al. 2001, p. 162). This research adopts the notion of competence as the 'ability to' think/analyze, do something, and relate to other people. These abilities belong only to individuals; an organization's abilities to perform depend on the individuals present in an organization at any given time.

Communication competence is regarded as important, and might be the most important competence in ISD (Cockburn 2001; Fitzgerald et al. 2002; White and Leifer 1986). Still, there is no agreement on what communication competence includes. The following discussion on being 'rational' in ISD might serve as an example (Fitzgerald et al. 2002). A rational developer acts "in a way that clients and users understand" (p. 126) while being rational in relation to formalized methods is often referred to as "doing the right thing in an efficient and logical way" (p. 125). Communication between developers and users will also involve domain competence. To ease communication in ISD processes it is important that developers have domain competence (Truex et al. 2000; Walz et al. 1993).

Competence and experience are related concepts. Experience might lead to competence, but this is not an automatic progression; as described in *Reflective Systems Development* (Mathiassen 1998), reflection might be needed as well.

Practice is often referred to as something distinct from both methods and competence. ISD often occurs in situations that are complex, uncertain, unstable, and unique (Mathiassen 1998). These situations are often laden with value-conflicts, in which individual actors and different categories of actors participate in the important and difficult work of creating a common understanding of both the task at hand and how to reach the stated goals (Mathiassen 1998). The developer uniquely enacts the method-in-action (Fitzgerald et al. 2002). While "it is more important to have specialized knowledge about problems and possible solutions than it is to have general knowledge on how to structure and conduct development processes" (Mathiassen and Purao 2002, p. 83). This paper maintains that practice is what actually happens in development, rather than what ought to or should happen according to the method or the competence deployed.

2.2 The relationships

The following focuses on the relationships between methods and practice, methods and competence, and competence and practice.

Methods and practice. Because formalised methods are devised to inform ISD practice, it is reasonable to expect that methods are widely used and that they contain advice on how actors should implement them in practice. However, method designers offer little practical advice on implementing methods in practice (Fitzgerald et al. 2002). And many organizations claim that they either do not use any formalized methods or that they use methods developed in-house (Huisman and Iivari 2002; Kiely and Fitzgerald 2003). This is surprising as it is widely believed that system developers' adherence to methods in ISD will benefit the organization (Huisman and Iivari 2002). On the contrary, even if method is one of the ISD discipline's key features, it is also probably the "true origin of its crisis" (Ciborra 1998, p. 8). Methodology can act as a social defence, undermine the learning process, and hinder creativity in ISD processes (Wastell 1996). It may also be questioned whether ISD methods really describe what happens in ISD practices (Truex et al. 2000). Their view is that if actors view practice through a method's concepts, things that happen only in practice are not noticed or registered unless they are formal concepts in the methodical arsenal. They therefore question the privileged view that ISD "is a managed, controlled process" (p. 60). Ciborra (2002) furthers this thought, introducing the term 'Bricolage' to describe what happens in an ISD process; actors creatively use whatever is at hand during development. Walz et al. (1993) observed an ISD team and were surprised by how difficult it was to communicate and to achieve a common understanding of the team's tasks during a requirements determination process. There are "discrepancies between the state of the art and the state of practice in using software engineering tools and methods" (Curtis et al. 1988, p. 1268 referring Zalkowitz et al. 1984). Problems and practices in ISD persist (Kautz et al. 2007). Although their focus is not specifically on the relationship between method and practice, it is obvious from their discussion that this relationship remains a complicated one in ISD. Methods

are still promoted as solutions to the ISD problems, but their deployment don't necessarily lead to successful systems. This might be because formalized methods seek to avoid relying on individual developers' abilities in ISD processes (Fitzgerald et al. 2002). In their method-in-action framework, Fitzgerald et al. (2002) discuss the ISD components in detail, while simply suggesting the components' relationships and influences.

Madsen et al. (2006) study the emergent method, which they define as "the actual unfolding development process and the activities, and applied method elements that comprise the process" (p. 226). Madsen et al. (2006) see the development process as a sequence of activities and argue that their emergent method goes beyond Fitzgerald et al.'s (2002) concept of method-in-action "as it places more emphasis on what actually happens over time than on the relationship between the prescribed and the actual" (Madsen et al. 2006, p. 226). They consider the actual development process a result of "a complex web and interplay of enacting and interacting actors and structures" (p. 226). Their analytical framework draws on three perspectives:

- The structuralist perspective relates to the structural characteristics of systems development concepts.
- The individualist perspective reflects how the individual developer influences and shapes the emergent method.
- The interactive process perspective counts for the method's dynamic emergence over time.

The result is "the emergent method and information system under development" (Madsen et al. 2006, p. 228). Madsen et al. (2006) assume a more holistic view of systems development than Fitzgerald et al. (2002), but they still concentrate their description and discussion more on the emergent method and its use in practice, and less on the relationships in play during ISD.

Method and competence. The relationship between method and competence has received considerable research interest. Some researchers state that methods are formalized competence. The advantage of this view is that competence is not needed to implement the actual methods in practice. Others state that developing information systems is both a technique and an art, or a creative process (Brooks 1987). If the developer follows the method strictly, it might preclude innovation (Fitzgerald et al. 2002) Because ISD is a creative process, it is important that the individual developers engage their competencies (Fitzgerald et al. 2002). Finally, Fitzgerald et al. (2002) state that developers learn by engaging in methods, but they do not discuss or incorporate this perspective in their method-in-action framework.

A possible clash between the Weltanschauungs of the method creator and method user will lead to the latter using the method in a way that differs from the creator's intentions (Jayaratna 1994). The use of methods will be influenced by both developers' competence and their views of software development (Cockburn 2001). The 'understood method' (Ørvik et al. (1999) can be achieved only by some kind of relationship between method and competence. Necco et al. (1987) comment on this relationship, stating that ISD's key factors are improved involvement and better personnel; method in itself does not suffice.

Published by AIS Electronic Library (AISeL), 2009

Competence and practice. Even though this relationship generates few hits in literature searches, the research literature directly or indirectly recognizes its existence. Madsen et al. (2006) stress "the importance of understanding the context, … the developers' preconceptions and actions and their interactions with other stakeholders, as well as the influence that these concepts have on the ISD process" (p. 227). To achieve this 'understanding' of what actually takes place in ISD processes, competence must relate to practice. But Madsen et al. (2006) do not discuss how actors use this understanding in an actual development situation. In studying how systems developers work in practice, Westrup (1996) suggests that their representations of organizations are actively constructed as rational, coherent, and fitting to computerization. Developers use their competence to analyse and form their understanding of the actual situation in practice. The reflective practitioner uses competence to reflect on practice, contemplating both how to proceed in practice and what learning might occur as a result of practical experiences (Mathiassen 1998).

Table 1 presents a summary of the research literature. The table is constructed to show the bi-directional relationships between method, practice, and competence. Descriptions of the relationship between method and competence were not found in research literature.

	Relationships		
	Methods	*Practice*	*Competence*
Methods		(Fitzgerald et al. 2002) (Huisman and Iivari 2002) (Kiely and Fitzgerald 2003) (Mathiassen and Purao 2002)	
Practice	(Fitzgerald et al. 2002)		(Fitzgerald et al. 2002) (Mathiassen and Purao 2002)
Competence	(Brooks 1987) (Fitzgerald et al. 2002) (Jayaratna 1994) (Madsen et al. 2006) (Mathiassen and Purao 2002) (Necco et al. 1987)	(Fitzgerald et al. 2002) (Madsen et al. 2006) (Mathiassen and Purao 2002) (Westrup 1996)	

Table 1: A summary of the reviewed literature

3 Research approach

3.1 Research method

Since the focus of this research is exploratory and descriptive, a case study approach is selected. The case study investigates "a contemporary phenomenon within its real-life context" (Yin 1994, p. 13) where "the investigator has little control over events" (Yin 1994, p. 1) and therefore cannot manipulate relevant behaviours. The research answers a how-question—in this case, how do competence, methods, and practice relate to and influence each other in ISD? This is in line with Yin's (1994, p. 6) criteria for a case study research strategy. Further, the case study approach gives the actors involved opportunities to describe their own and other actors' competence, methods, and practice in rich terms.

The unit of analysis is the organizational level. During the data analysis, it became clear that the research must also include the individual level to adequately understand and describe the relationships between competence, methods, and practice.

Hereafter, the developer organization is referred to as DeveloperOrg, while the user organization is called UserOrg. Data was collected through document study and semi-structured interviews, which were conducted in retrospect after the project's main part was implemented. There were six interviewees from DeveloperOrg: the project manager; the product managers responsible for the ERP system, the invoicing system, and the e-procurement system, respectively; and two domain experts engaged in the project. There were nine interviewees from UserOrg, including the project manager, the project coordinator, and the subproject managers. The interviews were tape recorded and later transcribed, and the transcriptions were sent to the interviewees for validation. The researcher received feedback on the transcribed interviews via e-mail. All of the email comments related to minor issues in the transcription.

The data analysis used grounded theory techniques, open, axial and selective coding (Glaser and Strauss 1967) as follows. The transcribed text was coded (open coding) based on the competence, methods, and practice of the seed categories' actors, and was therefore not fully open.

The relationships between the elements were coded (axial coding) using the open coding's coded text. Three different relationships emerged: competence/methods, competence/practice, and methods/practice. In axial coding, each element's influence on another element was also coded. This led to six different directions of influences between methods, competence, and practice.

Given the initial findings from the document study and interview data, two coherent reports were created describing the UserOrg and DeveloperOrg development stories, respectively. The reports were sent to the interviewees at the relevant organizations to validate the initial findings. An interview was then conducted with UserOrg's project manager to get feedback on the UserOrg report. Feedback on the reports was also received via e-mail from both UserOrg and DeveloperOrg. Again, the comments related to minor issues.

After receiving feedback from UserOrg and DeveloperOrg, additional coding (selective coding) was done and categories were combined into three topics that explained the relationships

Published by AIS Electronic Library (AISeL), 2009

and influences between competence, methods, and practice: "Intrinsic dynamic relationships", "Common understanding", and "Organizing vision".

3.2 Case background

UserOrg, a large (by Norwegian standards) local municipality, needed to replace its existing ERP system because the system vendor had announced that it would discontinue product support. UserOrg was searching for an ERP system that integrated accounting, budget, salary and personnel, invoicing, invoicing module feeding systems, and an e-procurement module.

UserOrg's IT manager organized the project internally; it included a steering committee with high-level officials to get easy access to decisions on financial matters, and project subgroups for each system module. Later, when the project entered the actual development phase, an informal project group was formed consisting of the IT project manager, a project coordinator, and all subproject managers. The externally hired project coordinator assisted the project manager, participated in project group meetings, and modelled work processes. The project manager and the project coordinator synthesized the different subproject groups' requirements specifications into one common tender document. Table 2 offers an overview of the project's main activities:

Time	*Activity*	*Comments*
Sept./Oct. 2001	UserOrg started internal process	Started as a substitution project
Feb. 21, 2002	Approved tender document	Developed process-oriented requirements specification
Feb. 28, 2002	Pre-qualification ended	Qualified two Developerorgs
March 2002	Demo-days	Two Developerorgs and UserOrg participated
April 10, 2002	Deadline for preliminary bid	Received two bids
May–Oct. 2002	Clarification of bids	UserOrg clarified bids with each of the Developerorgs
Oct. 2002	Final and best bid submitted	Reviewed by UserOrg
Feb. 2003	Contract signed	
May 2003	Development and implementation project started	Very close, active contact between UserOrg and DeveloperOrg
Jan. 1, 2004	All major ERP systems in production	Successfully implemented and set in production
Jan.–Sept. 2004	E-procurement and invoicing systems in development	Many new ideas and improvements
End 2004	E-procurement and invoicing systems in production	Successfully implemented and set in production
Spring 2005	All system modules in production	Systems development regarded as a success

Table 2: Development timeline

After the bidding process was concluded, the winning DeveloperOrg organized a project group consisting of a project manager, an ERP manager, an e-commerce manager, and implementation-process consultants. DeveloperOrg considered UserOrg a very important user of its system. Because UserOrg had very high domain competence in parts of the invoicing system domain and was to become the largest local municipality to install and use DeveloperOrg's entire ERP system package, DeveloperOrg took the opportunity to upgrade and improve its ERP system. The results of this project had positive effects on DeveloperOrg's market position in Norway.

4 Case description

The following case description reflects the three seed categories of competence, methods, and practice.

4.1 Competence

In analyzing the interview data, five competence categories were identified: domain competence, project competence, IS development competence, negotiation competence, and communication competence. Table 3 shows the similarities and differences between UserOrg and DeveloperOrg.

Competence	*UserOrg*	*DeveloperOrg*
Domain	Most actors had worked for many years in their specific domains	Actors had developed IS for local municipalities for many years
Project	Four of five central actors had previously participated in a large ISD project	Actors regularly worked on large ISD projects
IS Development	Little experience	Several actors had considerable experience and education in the field
Negotiation	Competence at management level and sought advice from a buying specialist	Competence at different levels, including the ERP-responsible
Communication	Good	Good

Table 3: Project competencies

Domain competence. One UserOrg actor was a leading domain expert in the invoicing system domain, and was specifically sought out by DeveloperOrg. Both UserOrg and DeveloperOrg were expecting high domain competence from each other and both report that their expectations were met. This shared domain competence seemed to make communication easier within the domains.

Project competence. UserOrg had changed ERP systems in the mid '90s. UserOrg's central actors and their project manager—who also managed the previous ERP project—had reflected on the earlier project's experiences and used their project competence to design this ISD project's main activities.

Development competence. UserOrg and DeveloperOrg had different development competencies. This led to different interpretations of certain incidents. The differences were especially visible in how they communicated in critical situations during the ISD.

Negotiation competence. UserOrg and DeveloperOrg were continually negotiating requirement specifications. Negotiation competence was therefore an important competence in the project, and was seemingly balanced between the two organizations.

Communication competence. Overall, both UserOrg and DeveloperOrg displayed high communication competence. In several incidents, however, actors in the two organizations failed to clearly communicate and this led to misunderstandings. This was especially visible during prototyping.

4.2 Methods

Neither organization used formalized ISD methods. Still, as table 4 shows, their project efforts included several method element categories: brainstorming, tender document development, demo-days, requirement and contract process, and ISD processes.

Brainstorming. The project's initial activity was to identify what the new system should do for the different departments at UserOrg.

Tender document development. The project coordinator modelled and documented the different departments' requirements specifications and—through "a process-oriented tendering process"—merged those requirements specifications into a complete tender document for the whole system.

'Demo-days'. Two pre-qualified development organizations presented solutions to a case that UserOrg designed. UserOrg's different subproject groups participated in the presentations related to their system modules.

Requirements and contract process. After the demo-days demonstrations, each subproject group separately continued discussions with the competing development organizations to clarify what was ready for delivery, what was in the pipeline, and what the development organizations were willing to develop to satisfy UserOrg's requirements. This activity produced the system's initial requirements specifications.

Methods/ method elements	*UserOrg tasks*	*DeveloperOrg tasks*
Brainstorming	Elicited initial requirements specification	
Tender Document Development	Created a common tender document to help select winning bid	Developed and submitted bid document
Demo-days	Clarified functions availability and used demonstrations to help select winning bid	Presented their solution and fielded questions from UserOrg actors
Requirement and Contract Process	a) Clarified bid documents and requirements for new system b) Negotiated terms and signed contract	a) Clarified bid document and requirements for new system b) Negotiated terms and signed contract
Development Processes	Dynamically elicited requirements specifications by: a) Performing and discussing daily tasks b) Testing prototype, making suggestions, and giving feedback c) Acting as a pilot user	Dynamically elicited requirements specifications by: a) Observing and discussing the UserOrg tasks with UserOrg b) Developing and testing the prototype c) Observing the pilot user d) Acting as middle-man

Table 4: Method elements used in the ISD project.

After selecting the winning bid, UserOrg requested that their requirements became part of DeveloperOrg's standard system. However, DeveloperOrg had to be careful not to introduce changes that would adversely affect their existing customers' system usage. It therefore handled the UserOrg request as follows: If DeveloperOrg developers found a proposed requirement beneficial, they would integrate it into the existing system. If proposed requirements did not fit into established plans, the developers first tried to find ways to fulfil the requirements directly within the existing system. If that proved impossible, they would look for a way to work around the requirement within the existing system. Requirements that remained unmet after these two steps were put on a prioritized list—according to usefulness and importance for DeveloperOrg—that was used in requirements negotiations.

The company that ultimately won the contract had decided early in the process that it would win and would make the delivery a success. The project was anchored in DeveloperOrg's top management; indeed, representatives of top management were members of the project group tasked with preparing the final bid.

> "But then, actually, then it had such high priority or focus with us that the final bid was prepared by the Managing Director. And I may say the working chairman of the board and me and the salesman at that time and another person." (ERP-responsible, DeveloperOrg)

Development processes. The initial requirements specification was a starting point for a further dynamic specification elicitation that occurred through close interaction between DeveloperOrg's domain specialist consultants and UserOrg's users. Based on their common suggestions, DeveloperOrg's module consultant sent suggestions in writing to DeveloperOrg's module-responsible. She then decided what to include in the requirements specifications and instructed the programmers accordingly.

In addition to this more formalized method element, the organizations used several informal method elements, including:

- DeveloperOrg's domain specialist, who was also the system-responsible, communicated directly with an actual UserOrg user on one side, and with DeveloperOrg's e-procurement system programmer on the other.
- DeveloperOrg's invoicing systems consultant (who was not a domain specialist) communicated UserOrg's requests and ideas to DeveloperOrg's ERP-responsible.
- The ERP-responsible communicated directly with UserOrg's representative.
- A DeveloperOrg domain specialist communicated directly with both a UserOrg consultant and DeveloperOrg developers/programmers (in cases of emergency). As the support-responsible from DeveloperOrg put it, much depends on the size of the problem:

  "If it is the calculations that fail completely, and 5,000 bills are to be issued tomorrow, we have to 'turn on the dime' and then just jump all formalities … try to get in the back door and solve the problem and get a new application to the UserOrg as soon as possible. So you are in the informal organization."

As the above examples show, even when there were agreed-upon methods for communications between UserOrg and DeveloperOrg, the methods were not always followed.

4.3 Practice

Table 5 describes two major activities performed in UserOrg and in DeveloperOrg. Different actors had different perspectives on the usefulness of demo-day presentations. According to the UserOrg project manager, having demo-days was a "conscious decision" with the purpose of exposing weaknesses in the system and determining what type of solution the two developer organizations could deliver.

UserOrg's project coordinator argued that it was important to balance power between the UserOrg and the DeveloperOrg in the ISD process; demo-days could help achieve this by making DeveloperOrg present solutions to problems UserOrg wanted solved. "Using a demo case gives UserOrg the lead," he said.

As the following quotes from demo-days participants show, not everyone viewed the activity as beneficial.

"… both developer organizations had too little time to prepare (for the demonstration of the case). … At the time of the demo, it did not benefit us much. It didn't." (subproject manager, UserOrg)

Activities	*UserOrg*	*DeveloperOrg*
Requirements elicitation	a) Produced demo-case and participated in demo-days b) Negotiated and employed contractual legal expertise c) ERP system development d) Pilot installation	a) Demonstrated their system based on demo-case b) Negotiated and strategically handled requirements specifications c) ERP system development d) Pilot installation
Staffing	a) Deployed domain competence b) A non-domain actor cooperated with DeveloperOrg to obtain domain competence	a) Deployed domain and development competence b) Gained domain competence through cooperation with UserOrg c) Chose a non-domain actor based on relationship to UserOrg

Table 5: ISD activities.

> "Use the exact data provided by them (the UserOrg) and try to reproduce the situations and demonstrate the processes they are looking for. As usual, you get too short a time. I remember that we did not get through it all." (representative, DeveloperOrg)

> "… it is often difficult to tell about the good news if you have to follow a big demo case from A to Z. … Such a demo may be very fragmented, making it difficult for the one who decides on what system to choose." (representative, DeveloperOrg)

Furthermore, information about the demo-days' purpose was presented only to UserOrg actors, not to those at DeveloperOrg. Still, as the interview data clearly shows, the demo-days' goals were neither understood by UserOrg's actors nor were they achieved.

When the requirements elicitation process began, both parties shared an interest in eliciting the best requirements. Later, DeveloperOrg used the requirements strategically during negotiations to win the contract and during the ISD processes.

UserOrg and DeveloperOrg deployed development and domain competencies in the ERP system development and the invoicing system's pilot installation. In the ERP system development, a high level of shared domain competence between DeveloperOrg and UserOrg made communication easy between the actors. DeveloperOrg basically handled the development technicalities, and the differences between DeveloperOrg and UserOrg in development competence (high and low, respectively) did not negatively affect UserOrg in this part of the ISD.

DeveloperOrg uses prototyping for major module revisions and to develop new modules, including (in this case) the invoicing system domain. UserOrg initially had high domain competence and low development competence, but the domain specialist went on sick leave shortly after the project started. UserOrg's substitute had low competence in both the development and the invoicing domain. When the invoicing system development started, DeveloperOrg had no domain specialists available. Staffing of the invoicing system ISD group was therefore partly based on the good relationship between some UserOrg actors and a DeveloperOrg consultant who had high development competence and low invoicing domain competence. In that situation, relationships were more important than domain competence. A main actor from each

organization cooperated in the development and improved their domain competence from low to high in the invoicing domain by developing the module together.

The differences in development competence turned out to be a challenge in the prototyping situation. However, DeveloperOrg was used to problems with prototyping and pilot installations in development projects:

> "We had some technical problems during the project. But we have that. We anticipate that we always have (problems) in projects. … we do not experience that as something critical". (Consultant, DeveloperOrg)

Reports from UserOrg contrast with this view. UserOrg actors generally felt that there were too many errors in the system during prototyping development. According to the DeveloperOrg representative, however, UserOrg's actors may not have understood the pilot user role or its implications. While the representative said that "it was entirely natural" that UserOrg members should be pilot users, it seems that "they were not conscious that they were pilot" on the invoicing system modules.

Indeed, when asked, one of the UserOrg actors said that he did not know that he was a pilot user. Differences in development competence between DeveloperOrg and UserOrg seem to lead to differences in interpreting the actual situation and the related activities.

5 Case analysis

5.1 Relationships

The three categories—competence, methods, and practice—not only relate to each other, but also influence each other. The following discusses and exemplifies the influences between categories for DeveloperOrg and UserOrg. The bi-directional relationships are illustrated in Figure 1.

Competence influences methods. An important DeveloperOrg objective was to win the contract for delivering the new system. As described (section 4), DeveloperOrg brought development and negotiation competence into play and used requirements specifications strategically as a method to conclude the contract negotiations.

UserOrg's project leader and most of the subproject leaders had previously acquired project competence in a large ISD project. This was clearly visible in how they chose and carried out brainstorming, tender document development activities, and the demo days.

Methods influence competence. UserOrg domain specialists often put considerable energy and time into creating their own requirements specifications prior to acquiring new systems. Given this, DeveloperOrg consciously used requirements specification elicitation as a method for getting good ideas about how to create functions or improve existing functions in their systems.

UserOrg did not have competence in prototyping as an ISD method. However, as they engaged in prototyping, their competence in both the domain area and in the ISD method increased.

Competence influences practice. DeveloperOrg's domain and technical competence let them tailor their bid to UserOrg's requirements specifications and thereby fulfil UserOrg's wishes within the project's technical and financial constraints. While doing this, they made sure that system changes had little or no adverse affect on the existing system users' daily and future practices.

Individual UserOrg users experienced increased system competence as the ISD led to some changes in how they used the new system.

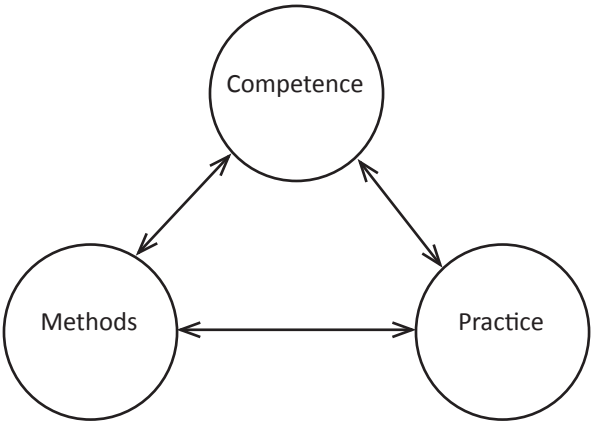

Figure 1: Bi-directional relationships between competence, methods and practice

Practice influences competence. Although it did not perceive a need for them, DeveloperOrg developed and installed special functions requested by UserOrg. UserOrg did not give any feedback to DeveloperOrg about these functions. Reflecting on the lack of feedback, DeveloperOrg's competence in requirements elicitation increased. Its developers will better scrutinize requirements elicitation and proactively seek out feedback from future client organizations.

Early in the ISD, UserOrg either received no installation manuals, or the manuals they did receive were insufficient for system installation. Once they requested and received better installation manuals, they made fewer mistakes and increased their competence in later installations.

Methods influence practice. In developing the e-procurement system, DeveloperOrg's representative consciously chose to use observation and discussion as a method (see Table 4 Development Processes section). This influenced how UserOrg and DeveloperOrg representatives worked together in practice. DeveloperOrg's representative reports that he was surprised by how UserOrg's e-procurement responsible used some functions very differently than how the system designer intended.

A UserOrg representative participating in the e-procurement activities said that the choice and use of the development method influenced her work in practice.

Practice influences methods. In emergency situations, DeveloperOrg's support responsible took shortcuts, using every possible way to fix a problem. In doing so, he disregarded the predefined methods for correcting system malfunctions. The support responsible was thereby able to solve emergency problems faster than if he had reported the error using the prescribed method.

Because some UserOrg representatives found the demo-days useless, UserOrg representatives will likely choose different methods to select the winning bid in future development situations.

5.2 The Relationships revisited

Influences described in the analysis and illustrated in figure 1 do not fully explain what happened in the ISD. A closer analysis reveals that the relationship influences often go via the third element. The following three examples—from DeveloperOrg's perspective—illustrate this finding.

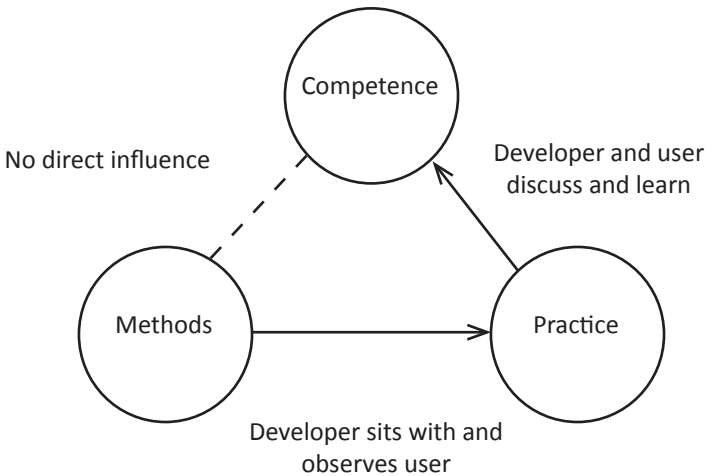

Figure 2: Method influences competence via practice

Method influences competence via practice. DeveloperOrg's e-procurement responsible chose observation and participation as the method for learning how UserOrg employees use the system to solve daily tasks (see figure 2). Using this method led to a change in the e-procurement responsible's domain competence and in system usage.

Competence influences practice via method. One of DeveloperOrg's actors had for many years studied how new requirements specifications affect existing systems (see figure 3). This actor's competence led to his developing and internalizing a method that influenced how he

worked in practice to ensure that requirements changes did not adversely affect existing system users.

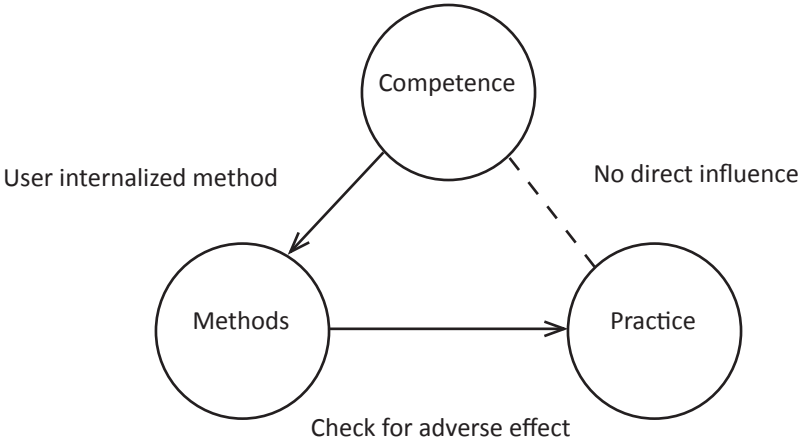

Figure 3: Competence influences practice via method

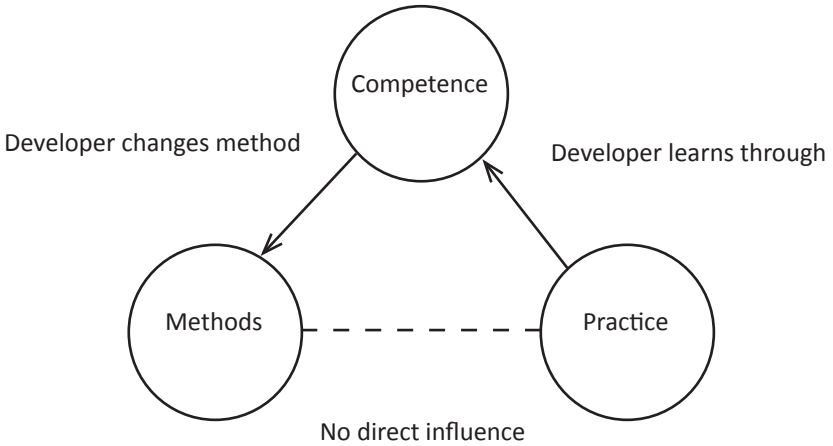

Figure 4: Practice influences method via competence

Practice influenced method via competence. DeveloperOrg agreed to develop some special functions for UserOrg. However, uncertainties about the actual usage of such functions within UserOrg led the DeveloperOrg consultant to use his competence to suggest ways (methods) that DeveloperOrg might meet this kind of challenge in future development projects (see Figure 4).

6 Discussion

6.1 Intrinsic dynamic relationships

As mentioned in section 5.2, influences between two elements can often be understood or explained only by actively involving the third element. These results both further and add details to Madsen et al.'s (2006) study of the emergent method. The term 'development-in-action' is therefore suggested to widen the emergent method's focus to recognize the role of competence and practice, which are at very least as important as methods in an actual dynamic ISD process (Mathiassen and Purao 2002). This change of focus challenges both the emergent method (Madsen et al. 2006) and the method-in-action (Fitzgerald et al. 2002) by describing, discussing, and understanding more specifically the influences that occur in ISD's intrinsically dynamic relationships.

Such relationships are dynamic throughout development. This fact is most clearly illustrated in the pilot installation processes, where actors used prototyping as an ISD method to refine requirements specifications and as part of the learning process.

6.2 Common understanding

The level of commonality between actors' competence, method knowledge, and activities in practice and the clarity of their communications influenced their level of common understanding.

When both DeveloperOrg and UserOrg had high domain competence (ERP development, section 4.3), the difference in development competence did not impact the ISD process. Several factors might explain this. First, the target system was a standard system. DeveloperOrg handled the technicalities, which did not negatively affect UserOrg. The similarities in domain competence made communication easy and clear among actors in both organizations and, in turn, eased the ISD process (Cockburn 2001; Fitzgerald et al. 2002; Mathiassen and Purao 2002).

A change of actors at UserOrg created temporary differences in domain competence (Invoicing module, section 4.3). One actor from each organization shouldered the main responsibilities for continuing this work successfully and increased their domain competence from low to high because they had good cooperation based on the good relationship they had established earlier in the project.

Different development competence existed between DeveloperOrg (high) and UserOrg (low) throughout the ISD (see section 4.3). DeveloperOrg actors and UserOrg actors did not initially share nor arrive at a common understanding of the ISD method, its use in practice, or the consequences of its use. This caused problems in ISD processes when using prototyping to develop the invoicing module. These problems might be explained as follows. While DeveloperOrg was accustomed to problems with pilot installations, UserOrg did not understand what a "pilot installation" meant. That is, the two organizations had a different understanding of the method's deployment (Ørvik et al. 1999). The DeveloperOrg actors constructed their

own representation of what happened (Westrup 1996), and did not heed UserOrg signals that there were problems until UserOrg representatives brought those problems to the attention of DeveloperOrg's top management. In the ISD situation, neither parties' actors understood the importance of the intrinsic relationships and were thus unable to actively clarify the situation before UserOrg escalated it. Ultimately, the problems were resolved through a dialog between top management at both organizations.

As this discussion shows, communication is an imprecise notion. General communication competence is insufficient; specific and shared domain and development competence can help actors obtain a clear and common understanding of what happens in the process (Walz et al. 1993).

As table 4 shows, negotiation competence played a particular role for DeveloperOrg actors, who used this competence to secure the contract and clarify the requirements specification. This did not negatively influence relationships between the two organizations. As the table also shows, both organizations had project competence, and interview data did not show specific problems related to the technicalities of running the project as such.

6.3 Organizing vision

While analyzing the case's data, the question of an organizing vision surfaced, inspired by Madsen et al. (2006). They suggest that "organizing around a vision emphasises the need for an IS project to be guided towards a desirable outcome rather than the blind pursuit of a planned result" (p. 236). This case study supports such a suggestion. At the same time, interview data makes this idea problematic in several ways. The organizing vision might be understood differently by different actors (Ørvik et al. 1999). Also, the organizing vision might get competition from other visions or goals in the process, or it might change dynamically in the ISD process. Such a change might not be communicated, or might be used tactically by one of the parties to obtain advantages. In addition, the way activities are carried out to reach the vision might clutter the vision, making it difficult for the actors to understand or navigate the processes. For example, both actors might want to develop a good system, but might disagree about what a good system is (as in Fitzgerald et al.'s (2002) discussion of what rationality means for practitioners vs. formalized methods). Given this, following an organizing vision (Madsen et al. 2006) might be as challenging to use as a development guide as blindly following a planned result.

As section 4 describes, another example of how challenging it is to reach an organizing vision is visible in the process of eliciting system requirements and creating a common understanding of and agreement on them (Mathiassen et al. 2000). This finding supports the Smolander and Rossi (2008) findings that, when creating an e-business or enterprise architecture in a large, complex ICT company, "the major problems to solve are organizational" (p. 36). Still, it ultimately seems that UserOrg and DeveloperOrg succeeded in agreeing on specifications through a dynamic learning and negotiating process. How can we explain that? Clarity of communication between the actors seems to be the best explanation. They worked together to reach a common understanding of both the situation and the specifications (Cockburn 2001; Walz et al. 1993).

Published by AIS Electronic Library (AISeL), 2009

6.4 Implications

One implication for theory is to emphasize development-in-action rather than focusing on method as in method-in-action (Fitzgerald et al. 2002) or the emergent method (Madsen et al. 2006). As Kautz et al. (2007) argue, there are persistent problems and practices in ISD independent of development trends or method use. They propose to focus on dynamic research questions related to diversity, knowledge, social structures, and an understanding of the underlying ISD problems. Research on the intrinsically dynamic relationships in development-in-action could further the understanding of the persistent problems and practices that Kautz et al. (2007) describe.

One implication for practice when designing and implementing an actual ISD process, is that it is more important to consider all three elements—competence, methods, and practice—and their intrinsically dynamic relationships rather than focus on methods alone. Both in educating developers and in the reflective systems development processes (Mathiassen 1998), the development-in-action focus can help actors understand, reflect on, and learn ISD processes.

A second implication for practice is that communication is a critical success factor. This is not a new point in the IS field. However, this research suggests that communication challenges in an actual ISD process relate to the degree of commonalities in the actors' competence, methods, and practice; in how they communicate about these factors; and in how they understand the relationships between them. In the prototyping process, for example, this research shows that big differences in competence, methods, and practice can lead to a less successful ISD process.

A third implication is that using a common organizing vision (Madsen et al. 2006) to guide development will be little more than words unless the actors share that vision, understand it in the same way, accept it, and act upon it. Because each actor in an ISD process might have his or her own agenda in addition to or as part of an organizing vision, the need for clarification is crucial. Understanding and using development-in-action might be one way to achieve such clarification.

7 Conclusion

The main contribution of this study is a deeper and more detailed understanding of the intrinsically dynamic relationships between actors' competence, methods, and practice in an ISD context. The understanding and description of these relationships furthers and details the method-in-action (Fitzgerald et al. 2002) and the emergent method (Madsen et al. 2006) and suggests development-in-action as a more suitable term and focus for an ISD process.

How development-in-action emerges in an ISD process depends upon how clearly actors in the process communicate. When actors have common domain and development competence and common organizing visions for the development, the intrinsically dynamic relationships seem to create clear communication and a more successful ISD process.

Recent research shows that large, formalized methods are seldom used in systems development. The research suggests that one reason for this is that formalized methods do not pay

enough attention to the individual developer's competence and his or her dynamic use of the method in practice during ISD processes.

This study suggests that it is not only the methods that "emerge" during ISD (Madsen et al. 2006). Both competence and practice also emerge through the interplay between them and the methods deployed in a dynamic ISD process. This emergence calls for further research to clarify what actually takes place, and especially how development-in-action emerges through an ISD process.

8 Acknowledgements

I am deeply indebted for the challenging and useful comments from Peter Axel Nielsen and the three anonymous reviewers. Thanks also to the "Othello" working group at IRIS'29 led by Lars Mathiassen. Thanks to Bjørn-Erik Munkvold and Tero Päivärinta for comments on an earlier version of the paper.

9 References

Avison, D. E., and Fitzgerald, G., (1995). *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill Book Company, London.

Bassellier, G. R., Blaize H., and Benbasat, I., (2001). Information technology competence of business managers: A definition and research model. *Journal of Management Information Systems*, (17:4): 159-182.

Brooks, F., (1987). No silver bullet: essence and accidents of software engineering. *IEEE Computer Magazine, (*April): 10–19.

Ciborra, C. U., (1998). Crisis and Foundation: An inquiry into the Nature and Limits of Models and Methods in the Information Systems Discipline. *Journal of Strategic Information Systems*, (7): 5-16.

Ciborra, C. U., (2002). *The Labyrinths of Information, Challenging the Wisdom of Systems,* Oxford University Press, New York.

Cockburn, A., (2001). *Agile Software Development*, Addison-Wesley, Boston, Mass.

Curtis, B., Krasner, H., and Iscoe, N., (1988). A field study of the software design process for large systems. *Communications of the ACM*, (31:11): 1268-1287.

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

Glaser, B. G., and Strauss, A. L., (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research,* Aldine, Chicago.

Huisman, M., and Iivari, J., (2002). The individual deployment of systems development methodologies. In: *CAiSE 2002*, Springer-Verlag Berlin, pp. 134-150.

Iivari, J., and Hirschheim, R., (1996). Analyzing information systems development: A comparison and analysis of eight development approaches. *Information Systems Journal,* (21:7): 551-575.

Iivari, J., and Lyytinen, K., (1998). Research on information systems Development in Scandinavia—Unity in plurality. *Scandinavian Journal of Information Systems*, (10:1&2): 135-186.

Jayaratna, N., (1994). *Understanding and Evaluating Methodologies. NIMSAD: A Systemic Framework,* McGraw-Hill Book Company.

Kautz, K., Madsen S., Nørbjerg, J., (2007). Persistent problems and practices in information systems development. *Information System Journal*, (17:3): 217-239.

Kiely, G., and Fitzgerald, B., (2003). An investigation of the use of methods within information systems development projects. In: *Proceedings of IFIP WG 8.2 Conference, Athens Greece*.

Kraiger, K., Ford, K., and Salas, E., (1993). Application of cognitive, skill-based and affective theories of learning outcomes to new methods of training evaluation. *Journal of Applied Psychology*, (78:2): 311-328.

Lee, D. M. S., Trauth, E. M., and Farwell, D., (1995). Critical skills and knowledge requirements of IS professionals: A joint academic/industry investigation. *MIS Quarterly*, (19:3): 313 - 340.

Madsen, S., Kautz, K., and Vidgen, R., (2006). A framework for understanding how a unique and local development method emerges in practice. *European Journal of Information Systems,* (15): 225-238.

Mathiassen, L., (1998). Reflective systems development. *Scandinavian Journal of Information Systems*, (10:1&2): 67-118.

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., and Stage, J., (2000). *Object-Oriented Analysis and Design,* Forlaget Marko, Aalborg.

Mathiassen, L., and Purao, S., (2002). Educating reflective systems developers. *Information Systems Journal*, (12): 81-102.

Marcolin, B. L., Compeau, D, R., Munro, M. C., and Hupp, S. L., (2000). Assessing user competence: Conceptualization and measurement. *Information Systems Research*, (11:1): 37-60.

Necco, C. R., Gordon, C. L., and Tsai, N. W., (1987). Systems analysis and design: Current practices. *MIS Quarterly*, (11:4): 461-478.

Schaumbach, T. P., (1994). Maintaining professional competence: an evaluation of factors affecting professional obsolescence of information technology professionals. Ph.D. dissertation, University of South Florida.

Smolander, K., and Rossi, M., (2008). Conflicts, compromises, and political decisions: Methodological challenges of enterprise-wide e-business architecture creation. *Journal of Database Management,* (19:1): 19-40.

Truex, D., Baskerville, R. and Travis, J., (2000). Amethodical systems development: the deferred meaning of systems development methods. *Accounting Management and Information Technologies*, (10:1): 53-79.

Walz, D.B., Elam, J.J. and Curtis, B., (1993). Inside a software design team: Knowledge acquisition, sharing, and integration. *Communication of the ACM*, (36:10): 63-77.

Wastell, D., (1995) The fetish of technique: methodology as a social defense. *Information Systems Journal*, (6:1): 25-40.

Westrup, C., (1996). Transforming organizations through systems analysis: Deploying new techniques for organizational analysis in IS development. In: *Information Technology and Changes in Organizational Work,* W. J. Orlikowski, G. Walsham, M. R. Jones and J. I. DeGross (eds.), Chapman and Hall, London, pp. 157-176.

White, K. B., and Leifer, R., (1986). Information systems development success. Perspectives from project team participants. *MIS Quarterly,* (10:3*)*: 214-223.

Yin, R. K., (1994). *Case Study Research: Design and Methods* (2nd ed.), Sage Publications.

Zalkowitz, M., Yeh, R., Hamlet, R., Gannon, J., and Basili, V., (1984). Software engineering practices in the U.S. and Japan. *IEEE Computer*, (17:6): 57-66

Ørvik, T. U., Olsen, D. H., and Sein, M. K., (1999). Deployment of system development methods: Exploring paradigmatic mismatches, evolution and challenges. In: *System Development*, J. Zupancic, W. Wojtkowski, W. G. Wojtkowski and S. Wrycza, (eds.), Kluwer Academic/ Plenum Publishers, New York, pp. 19-31.