December 2004

The Role of Methods in Software Process Knowledge Creation.

Mikael Schoenstroem
Lund University

Follow this and additional works at: http://aisel.aisnet.org/pacis2004

Recommended Citation

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
The Role of Methods in Software Process Knowledge Creation.
Mikael Schönström
Informatics, School of Economics and Management, Lund University
SE-223 63 Lund, Sweden
mikael_schonstrom@hermes.ics.lu.se

Abstract
Software development is a social process in which innovation and knowledge creation are important factors for project success. How knowledge in large software organizations is created has been scarcely researched despite extensive research regarding knowledge creation processes in general. How methods and practice interact in knowledge creation processes in large software organizations has not been thoroughly studied. In the software development literature it has been argued that methods play a role as a framework for communication and experience capturing, thus facilitating re-use and creation of new software process knowledge. Another research stream presents a contradictory picture and argue that methods are seldom used in practice, and that their positive impact on software development has been exaggerated. There is however a common argument in the IS literature that most organizations report that they use methods and spend considerable resources on their development and implementation. This paper discusses knowledge creation in software development organizations, and what role methods play in a knowledge creation process consisting of method and practice. Drawing on data from an observational study the paper presents and discusses a number of lessons regarding methods’ role in knowledge creation in software organizations.

Keywords: Knowledge management, knowledge creation, software development, method, participant observation.

1. Introduction
The importance of knowledge creation in high-tech firms has been stressed in both knowledge management and strategic management literature (e.g. Nonaka 1995; Drucker 1995). Telecommunication companies like Ericsson, Nokia, Lucent are often seen as key drivers of technological innovation because telecom systems have fundamental effects on society and R&D investments. Much of the features provided by telecom systems are today software based. Software projects in this type of companies are often extremely large and complex where much new knowledge is created. In these organizations skilled project members are necessary in order to successfully manage and coordinate the web of development projects.

Despite the importance of software development in telecom companies, research that has focused on how software projects create new knowledge in these companies has been limited. Research on knowledge creation as phenomena is however not new and several models for knowledge creation have been proposed. The theories presented by Nonaka and Takeuchi (1995) and Cook and Brown (1999) are some of the best known. Recent research on learning in project environments have also suggested models for how learning is supported and what mechanisms that enable learning (Bresnen et al. 2003; Schindler and Eppler 2003). These theories however, do not give a full picture of how software projects create new knowledge. The special characteristics of software projects e.g. project size, product complexity, the use
of different support tools and methods create a complex context, which is important to consider when studying knowledge creation processes in software organisations. Models that explain knowledge creation in software development contexts are uncommon, even if some SPI (software process improvement) related literature and method literature have discussed knowledge and learning around software processes (see e.g. Basili et al. 1992; Basili 1993; Baskerville et al. 1998; Mathiassen et al. 2001).

Models that describe how software projects create new knowledge are needed because of the special contextual characteristics of software projects. An increased understanding for knowledge creation processes in software development projects would improve managers’ possibilities to manage and enable knowledge creation in software projects.

In the information systems development (ISD) literature the systems development method have been suggested to be a form of knowledge (e.g. Rossi et al. 2000; Hirscheim et al. 1996). A preliminary assumption could be that software projects create new knowledge by using software methods in their project work and then feedback new experiences to the method. A significant amount of resources are spent on methodologies in software companies and methods are a key research area. Contradictory though, much research indicates that software development methods are seldom used in practice (Fitzgerald 1997; Truex et al 2000; Baskerville and Stage 2001; Iivari and Maanasaari 1998). The contradictory information regarding the use of software methodologies and the suggestion that methods is a form of knowledge creates a need to understand the role of methods in knowledge creation processes within software organizations.

The purpose of the paper is to explore how methods and project practice interact to create new knowledge in software development organizations. Using a study based on participatory observations the paper presents a number of key issues with methods in practice and their potential to be part of a software organization’s knowledge creation process. The remainder of the paper is organised as follows: in the next section knowledge management theories related to knowledge creation is presented and discussed. In section three the research method is presented which is followed by a presentation of the case and the findings in section four. Discussion and conclusions are presented in section five.

2. Knowledge, knowing and knowledge creation in software projects

2.1 Knowledge and knowing

The knowledge management literature suggests many different views and types of knowledge (see e.g. Alavi and Leidner 2001), but two major perspectives on knowledge can be discerned, that of knowledge and that of knowing (Orlikowski 2002; Cook and Brown 1999). The first assumes knowledge as embodied and possessed by an individual. The other, knowing, is used to explain knowledge in action. In recent years it has been argued in the knowledge management literature that knowledge should be seen as something we do rather than something we possess (Blackler 1995; Cook and Brown 1999; Orlikowski 2002). The basic reason for an increased interest in knowing is that the epistemology of possession (knowledge) cannot account for knowledge as part of action, thus a broader perspective on knowledge is required. In this perspective it is suggested that the term knowing is complementary to that of knowledge. Knowledge is something that is static, possessed by an individual, e.g. knowledge about the Java programming language and its syntax. Knowing is actionable knowledge, knowledge as something we do. In the Java programming example it would be the actual coding action, where the software developer uses his knowledge in Java to code a
new function that is part of a larger system. In this situation the developer uses enacted knowledge, the developer knows how to use and interact with development tools, compilers, test cases to transform a requirement into executable software. Knowing is a valuable concept because it enables us to study knowledge used in action, and how action is a form of knowledge (Orlikowski 2002). Cook and Brown (1999) argue that it is important for our understanding of learning and innovation to view knowledge as a tool of knowing within situated interaction with the social and physical world. The reason is because the interaction is a linking of knowledge and knowing which makes the “generative dance between epistemologies” possible, which is the source for innovation. The interaction creates both new knowledge and new types of knowing according to Cook and Brown (ibid).

2.2 Knowledge creation in software organizations

Knowledge creation has been presented as one of the most important processes to knowledge intensive organisations for creating competitive advantage (e.g. Nonaka and Takeuchi 1995; Teece et al. 1997). Language is a major component in human knowledge creation since it is via the language we give meaning to words, which we use to represent objects and feelings in our environment. In knowledge intensive organisations that work with complex problem solving the language of a community becomes crucial to knowledge creation. Boland and Tenkasi (1995) suggest that knowledge creation in knowledge intensive firms is communication within and between a firm’s communities of knowing. A community of knowing is a community of specialized knowledge workers that has developed language and solid perspectives about their environments and thought worlds. They argue that perception is only accomplished through a perspective. Knowledge creation within a community is a process of problem solving, and in this problem solving process the vocabulary and theories are extended and refined to embody that perspective. Strong perspectives are important in knowledge creation because without strong perspectives a community cannot make a difference between what is important and what is not important (Boland and Tenkasi 1995). A method defines language and concepts, and therefore supports the creation of strong perspectives. In this paper the term method is used to denote a “recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems.” (Maddison 1983, cited in Avison and Fitzgerald 1995). It is a broad definition but an example of a popular method used today in the software industry is RUP (Rational Unified Process), which not only covers software engineering principles but also project management activities.

How does knowledge and knowing interact to create new knowledge and knowing, and how could these two types of knowledge interact in a software project to create new knowledge? Cook and Brown (1999) suggest that in the interaction between knowledge and knowing knowledge gives structure and discipline to knowing. Knowledge is used in knowing, and the interplay between these different “types” of knowledge create both new knowledge and knowing. They use the example of flute makers to explain how this work. The knowledge of the master flute maker is not transferred to the apprentice rather it guides the apprentice in its interaction with the world, so that the apprentice generates new knowledge. Thus, both the apprentices’ earlier tacit and explicit knowledge is used in the work and is further being influenced by the master’s tacit and explicit knowledge to generate new knowledge and knowing (see Figure 1).
In a software development context explicit and tacit knowledge would be the knowledge software developers and their team holds about processes, tools and relevant computer science techniques (database technologies, programming languages etc). Knowing is their ability to interact with the world (clients, vendors, colleagues, development tools) to develop a software product which meet the users’ expectations. Thus in this context tacit and explicit individual and group knowledge is used in the problem solving activities of software development, and in these activities new ways of knowing are created as well as new tacit and explicit knowledge. Explicit knowledge in a software development project would be the processes it uses; in-house developed processes or standard processes such as RUP. Tacit knowledge would be related to experiences about how these processes work in a real project. Thus both new explicit and tacit knowledge would be created while the project works with a particular method. When the project interacts with the world using the method as guide new knowing and knowledge is created. In the interaction between methods and practice, methods guide the actions of the project.

Knowledge “…gives particular shape, meaning, and discipline to our interactions with the world.” (Cook and Brown 1999). This is the potential role software development methods play in software development projects. By giving structure to the development process the method can support the re-use of existing knowledge and thus support the creation of new knowledge. Hirschheim et al. (1996) argue that: “The value of methods and tools, and their conglomerates—methodologies—is that they embody practices and cognitive frames that can be taught, shared and refined over continued trials. Methods can thus be conceived as consisting of directions and rules of action (stocks of knowledge) according to some systematic ordering…” . Knowledge creation in software development firms can thus be viewed as an interplay between the method (knowledge) and the project work (knowing). Rossi et al. (2000) argue e.g. “Thus, methods need to be maintained based on reflections from practice, transforming tacit knowledge into explicit knowledge.” An efficient interplay and feedback between practice and the method is then needed to make this interplay an important part of the learning process for software organisations.

However, there is a stream in the methods research literature that argue that it is problematic to use methods since changes in the development context makes planning foresight very
difficult (Truex et al. 2000; Cugola and Ghezzi 1998; Fitzgerald 1996). Further, research on development methods has found that they are seldom used literally, in practice, and if they are used they are adapted (Iivari and Maansaari 1998; Fitzgerald 1997). Baskerville and Stage (2001) suggest e.g. that software projects rarely uses methods but presents the idea that software development is an emergent practice, where fragments of methods are used. If this were the situation, it would imply that it is less valuable to view the interplay between methods and practice as an important knowledge generating process for software development organisations. Due to the anti-method literature we cannot take for granted that the interaction of methods and practice is an important knowledge creative process to a software development organisation. It is however a fact that methods are still developed in organizations and many organizations say that they use them. The main aim of this paper is therefore to explore how methods and practice interact and what role methods have in the knowledge creation process in a software development context. An observational study is presented in section four that gives further insight on how methods and practice interact in a software development organization, but first some words about the research approach.

3. Research Methodology
Participant observation was chosen as research method, because it allowed rich study of the phenomena of the company’s product development context and how it used its methods. Engaging in the daily activities of the software development project and collecting ethnographic evidence was valuable in studying the relationships, communication patterns and information exchange within this specific social group. The specific study followed a method development project in which I took active part as project member, contributing with method expertise.

Walsham (1995) argue that none of the roles an observer can play in a research setting should be viewed as an objective reporter since both data collection and analysis involves the researcher’s own subjectivity. He goes as far as mentioning that even outside observers could be viewed as action researchers as they influence what is happening in the domain of action. The study involved observations during a method development project within a large software development organization. My involvement in the project can be considered close to action research but some characteristics of the research make it problematic to categorize as action research. First, the research did not involve a joint collaboration between the university and the company. Second, from the company perspective my involvement in the project was not as a researcher but as project member, even if the other project members were aware of my dual role as researcher and project member. Third, the research process did not follow the action research process as described by Baskerville (1997, 1999). Thus I have chosen to categorize the research as participant observation.

To give an “objective” picture of what happened is a major issue with observations when writing up the report (Walsham 1995). Said Bogdewic (1999) “How is it possible to stand back and observe that of which you are also an integral part?” However, participant observation have some strengths, because instead of only relying on what people say in interviews you can actually observe what meaning people give certain actions in a particular setting. Participation made it possible to study the discourses of software development in a particular context. It was valuable because while the researcher and observed spoke the same language certain concepts acquired different meanings within the specific social group. The main sources of data were: ethnographic material collected in the form of filed notes, e-mail conversations, internal company documentation, meeting minutes and project documentation.
The observations were made from October 2002 to June 2003. The observations focused on how existing methods were used and how these were used in practice. I studied the projects’ existing work methods to get a better understanding for what characterised software development work, and what role methods played. I was a complete member of the group, and I participated in meetings and workshops in a method development project, called BETA (in this paper). BETA had seven members, four software project managers, a quality manager, a line manager and myself.

4. Case Findings

4.1 The Case Company
The case company is a subsidiary within a global telecom group. The company develops and sells mobile platforms to mobile phone manufacturers. The company’s headquarters is based in Sweden and it has offices in UK, USA, Japan and Norway. The company has about 1000 employees. The company offers complete 2.5G and 3G technology platforms to manufacturers of mobile phones and other wireless devices. The company’s technology platforms include complete component specifications, printed circuit board layout and software. The company also offers support and service in customising the platforms.

Four product development programs within the company were run concurrently at the time of the observations. Each program consisted of projects, and each project consisted in its turn of sub-projects. Thus, three project levels could be identified. The program level was organised from a company perspective and was run during 2-3 years. The software development project I worked with was a project with 150 developers, testers and project managers, developing new software modules for a 3G cell phone. Further on in this paper the development project is called DELTA, the company ALFA, the method development project BETA and the group ALFA belonged to is called EPSILON.

The background to BETA was ALFA’s long-term objective to improve control and efficiency of their software development projects. The method development project, BETA, was launched with the goal to create a common project method for a product development division within ALFA, in which DELTA was its most important development project. The purpose was to improve the software project management process by the use of a common method. I will further on use the term project method because the developed method in BETA had a focus on software project management, i.e. coordination and tracking guidelines for the software development projects.

4.2 The Work Environment
DELTA’s work environment was both complex and volatile. Immense numbers of issues and tasks had to be dealt with on a daily basis, which had not been anticipated during the planning phases. The work environment gave a very intense impression due to constant re-prioritisation, co-ordination, problem solving and short schedules with numerous deadlines. Software development methods were not used to any significant degree in DELTA, despite the availability of many methods, both commercially like RUP (Rational Unified Process) and in-house developed methods for project management and configuration management. The sentiment among many managers and employees were that the situation would improve if the project used common methods. However, the existing methods were extensive and to use them by the book was not considered feasible. Many feared that project management and administration would become unnecessary tedious and time-consuming if they were followed.
Existing methods had also an extensive focus on documents, specifications, plans and so forth, which resulted in time-consuming document updating.

In the company no less than 76 different methods had been developed and documented. Everything from resource allocation to how to simulate certain technical functions had been specified. The abundant amount of methods created a sentiment among some project members that the company did not need yet another method. The method situation resulted in that few methods were used at all. The limited use of common methods had though created a situation where synchronisation between projects had become a complicated affair. Despite the general critique regarding methods some managers believed that a common project method would improve the situation. It was urgently needed since the quality of the developed products tended to decrease and project members risked being unnecessary burned out because of double work due to inefficient planning and coordination. The lack of common methods was also estimated to limit learning efforts since project experiences were not captured in a planned and organized way. Thus, DELTA’s line organisation set up a project (BETA) with the goal to develop a common project method, focusing on a few mandatory milestones and criteria. The idea was further not to develop a “new” method but to re-use existing concepts and methods as much as possible and to adapt them to the current situation and to make them common. The common method was supposed to create a basic structure to enable better co-ordination between projects and to specifically facilitate the implementation of important CM (configuration management) activities, which were believed to increase product quality.

4.3 Methods and Practice
The project method developed in cooperation with DELTA focused on a number (7) of common milestones, to enable synchronisation between projects and sub-projects. BETA re-used concepts from corporate models and methods; the concept of milestone was e.g. taken from the company’s generic method for project management. We used existing parts from both corporate methods; best practices and local experiences then re-framed these to fit the scope of the target organization and its projects. Despite the use of existing knowledge it took time to develop and implement the method due to communication problems between some groups. Units had slightly different views and perspectives on software development, and the differences were large enough to create inefficiencies when communicating in workshops how things were done today and how they should be done in the future. BETA developed the new method in close co-operation with DELTA but still the implementation required learning efforts from DELTA’s project managers. Different agendas and experiences among the project managers made it in the beginning hard to promote the use of the “new” common method. However, during implementation and the first usage of the method it was estimated that synchronisation between projects improved and that important issues like lack of resources prior to project execution were raised in a more visible way than before.

Feedback on the developed method was rare unless the software project managers were specifically asked for feedback. The limited feedback may be explained by the situation, where many things needed to be done at once and different prioritisations and agendas competed for attention. However, also the method development unit within ALFA declared during a meeting that it was difficult to get feedback from the software practitioners. The reason mentioned was that the projects did not have the time, or the feedback routines or lacked the will to make the effort to communicate.
4.4 Knowledge, Knowing and Knowledge Creation in the Organization

What constituted knowing in the organisation? As one person came to me one day and said, “there is a lot of talk that we don’t have a method, but you see even if things seem to be messy and uncoordinated we still manage to get things done and to deliver. And we have been able to meet major delivery dates. So there must be some type of process in place”. Thus there existed an ability within the organisation to get things done even if a method were not officially used. It was very difficult to observe what this ability was based on. But the project’s geographical concentration to one location clearly facilitated co-ordination and ad-hoc meetings. The project and its sub projects could swiftly re-prioritise actions. The project had also been organized for a long period of time and most of its members had been working from its start, 1,5 years earlier. Most people knew each other fairly well, and had knowledge about roles, responsibilities and who had what knowledge.

A close group certainly helped communication between the projects. Other abilities observed that related to knowing was the ability to identify and solve issues quickly. DELTA’s project management were quick in finding out how an issue related to other issues and how it should be prioritised. There was a constant re-evaluation of the situation, and re-prioritisations of tasks on all levels involved, which contributed to the chaotic impression of the project. The ability to handle this messiness and to take quick decisions was one of the most important abilities DELTA had besides from its deep technical knowledge. The case showed that the project withheld process knowledge even if official and documented methods were not implemented or used.

In DELTA knowledge creation in practice regarded mostly the product under development, how should the product work, what parts were most critical to the client? Issues regarding testing and integration were numerous. The product complexity required much problem solving and thus knowledge creation. In this environment the knowledge used related to knowledge about basic technologies in computer science and telecommunication used to develop the product. This knowledge was considered to be most important. Process knowledge was not valued to the degree of technical knowledge, but this was something BETA was supposed to change. The limited use of methods resulted in that explicit process knowledge were only used to a limited degree.

In the studied case knowledge about processes and management of development projects were instead created during the workshops held by BETA to create the project method. In the workshops project managers and other key personnel had to begin to reflect over what they were doing and how they were doing it. BETA discussed project problems, what did the corporate model recommend, and above all, people from different competence areas met for the first time together to discuss management issues. Usually during the regular project management meetings in DELTA that where held once a week all disciplines (like product management, test management) were not represented and the issues discussed related more to technical problems, and the meetings had an ad-hoc problem solving focus. It was striking how unaware project members were about each other’s projects problems and perspectives regarding the way of working. Test personnel had one view, software developers a second, the project manager a third and the line managers a fourth. To agree on project phases took quite some time and just to churn out the meaning of a milestone took one workshop of two hours, despite the fact that the concept of milestone had an official description in the company’s well-documented project management method.
5. Discussion of the Findings and Concluding Remarks
The sporadic use of methods and the limited communication between method developers and projects in ALFA can be explained by two factors; the limited trust practitioners had for ALFA’s methods developers and the volatility in its projects. Software projects are different and some projects, like DELTA, operate in an extremely changing environment. DELTA developed an entirely new product, where both new hardware and software was developed. Not even the international standards for 3G systems were set. The prerequisites for the project constantly changed. When a plan was set, something happened that changed the project plan e.g. a client that re-prioritised the development of a feature or a technological issue that had not been anticipated. Planning activities specified in the methods could impossibly foresee all changes and dependencies between parts of the product, organisational units and their different agendas. Thus, the more uncertain the project environment is the less useful are detailed methods and the more important are frameworks or models that enable a platform for communication, but leave enough room for flexibility. In the turbulent environment of DELTA knowing played a key role when acting on chaotic change. Project managers constantly had to re-prioritise actions, resources and so forth, methods could only give marginal support in these cases. When methods were used they were at best used as reference and the experiences created in the project were seldom discussed in relation to the methods in an organized way so that the new insights could be brought back to the organization. In order for new process knowledge to become explicit group knowledge it has to be fed back so that the method can be updated. It was complicated in the studied case to view methods as a form of knowledge which disciplines knowing according to Cook and Brown (1999), because methods were simply only sporadically used in the everyday work. The interaction between method and practice (the use of the methods in the projects) was limited and can therefore not explain knowledge creation within DELTA.

The method development activity on the contrary created new knowledge and knowing regarding software development processes. To discuss management issues and to gather key people from different backgrounds proved that new insights about the development process were made. Different communities (e.g. programming) learned about other communities’ (e.g. integration and verification) way of working which created a better awareness for the whole process. Managers and experts used their tacit and explicit knowledge to reflect over their work processes in an organized and planned way. This interplay between the explicit knowledge documented in the method and the tacit and explicit knowledge of the practitioners increased project members knowing regarding the use of development methods and their process knowledge in general. In the studied case project managers improved their actions regarding e.g. staffing issues and coordination between projects due to the method development activity. Existing common methods facilitated communication regarding management issues, since they provided a platform, which were used as point of departure when discussing process issues. The intense discussions in the workshops enabled the project members to create a shared view of the development process that had not existed earlier despite the existence of different methods. Within DELTA three different knowledge areas could be defined; technological knowledge, organizational knowledge and process knowledge. The model created by BETA supported DELTA’s creation of process knowledge and knowing. BETA was however an isolated activity and in order to develop new knowledge about processes across organizational units a continuous reflection and discussions should be planned and supported within the whole organization (Schindler and Eppler 2003).
Three major findings can be suggested based on the study. First, knowledge creation in software development projects cannot be explained by the interaction of methods and practice, simply because methods are seldom used actively in software development projects (which is in line with earlier research on methods use). Even if fragments of a method are used the feedback loops of new experiences are usually missing. Second, to organize and plan everyday work in software development projects “tacit” methods based on tacit knowledge and knowing is used rather than formal documented methods that focus on explicit knowledge. Managers and method developers who work with projects that operate in an ever-changing environment should therefore pay increased attention to knowing and try to understand how knowing is supported and developed in software development projects. Third (and foremost), the method development activity generates new software project process knowledge. Common documented methods should therefore be used in project reviews to reflect over how did the project work, what went well and how can we improve our practice in the next project? To improve software projects’ process knowledge method development activities or reviews should be planned and organized by the line organization. In these reviews existing methods should be used as a point of reference and as a learning enabler around software processes.

The purpose of the paper was to study the role of methods in knowledge creation in a software development organization. Based on an observational study the interplay of method and practice was discussed and a number of lessons on knowledge creation in software development organizations were presented. The first two findings confirm earlier research on methods use; that methods are not used literally in practice and that developers earlier experiences are rather used than a formal method. However, these findings are presented from using an additional perspective, that of knowledge creation and knowing. The third finding suggests new insights regarding the role methods play in a knowledge creation process, within a software organization. It showed that the method development activity supports software project managers’ creation of new knowledge and knowing regarding software development processes. The method is important as a framework because it function as a common platform within projects and between projects, thus facilitates communication and understanding. It gives a structural framework for how to view, think and refer to events in the development process. The method is the structure (knowledge) used as a starting point and when combined with existing individual and group knowledge and knowing in a method development activity it enables the creation of new knowledge and knowing regarding software development processes.

Continuous research on knowing in software development organizations is needed, and for future research it may be necessary to expand to more psychologically based theories, like activity theory. Activity theory may shed some new light on the knowing aspects of software development as it takes consideration to individuals’ interaction with the world and broadens the scope to see the interaction as situated in the context of a historically developed collective praxis, an activity system.

References


