SOFTWARE PROCESS IMPROVEMENT THROUGH KNOWLEDGE MANAGEMENT

Sven A. Carlsson
Informatics, Jönköping International Business School
P.O. Box 1026
SE-551 11 Jönköping, Sweden
Tel: +46 36 157504
sven.carlsson@jibs.hj.se

Mikael Schönström
Informatics, School of Economics and Management, Lund University
Ole Römers väg 6
SE-223 63 Lund, Sweden
Tel: +46 8 56863444
mikael_schonstrom@hermes.ics.lu.se

ABSTRACT
Software Process Improvement (SPI) is an approach to systematic and continuous improvement of a software development organisation's ability to develop quality software. SPI requires improvement trough learning and knowledge creation around software development processes. Knowledge management (KM) is therefore an important component of a successful SPI effort. Many SPI programs are often based on normative models that fit badly with the unique and changing environment of software development projects. This paper presents and discusses two different approaches for knowledge management (KM) in software development organisations. Using the two studies we give recommendations for KM in software development organisations and how SPI can be supported by KM. The first study discusses a project that started with a common strategy to software process improvement—the Experience Factory—and later changed to a networking strategy. The second study discusses software development methods as knowledge enablers and their role in an SPI effort. One finding is that codified knowledge that lacks context is difficult to exploit in a volatile product development environment. A second finding suggests that there must be a balance between exploration and exploitation of knowledge as both are of importance to the software development process. A third finding suggests that methods as knowledge enablers play an important role in SPI.

1. INTRODUCTION

Based on the firm's strategic focus and resource base, Løwendhal (2000) identifies three generic types of knowledge-intensive (KI) firms. One type has a strategic focus on creative problem solving and has a team-based (individual and collective) resource base. An example of this type of KI-firms is software development firms as these firms create teams and projects with highly skilled professionals in order to solve very complex tasks (Newell *et al.*, 2002; Iivari and Linger, 1999; Quintas, 1996).

Software development is a type of NPD process. NPD is a continuous process of knowledge creation and knowledge use, in which the organisation is adapted to its changing environment of markets and technologies (Dougherty, 1992; Leonard-Barton, 1995). Nonaka and Takeuchi (1995) say it most elegantly: "Organizational knowledge creation is like a 'derivative' of new-product development. Thus, how well a company manages the new-product development process becomes the critical determinant of how successfully organizational knowledge creation can be carried out." In many high-technology firms software development involves extensive research and development activities where much knowledge creation and exchange takes place (Boehm and Basili, 2000). Furthermore, software development firms derive much of their competitive advantage from new products and services. In general, these firms are also under growing pressure to develop new products faster and at lower cost, without quality loss. In order for these firms to remain competitive in a global economy, they have to be effective and efficient in knowledge management around software development processes. Software process improvement (SPI) has received increased attention by software development firms in recent years as the software development *process* is important from a competitive advantage perspective. Within SPI, KM has emerged as an important factor because software development is knowledge intensive work where knowledge creation and sharing play a central role (see e.g. Mathiassen and Pourkomeylian, 2001).

Numerous strategies and approaches to KM have been discussed during the last decade. This paper presents and discusses two different approaches for knowledge management (KM) used in software development organisations. Using the two studies we give recommendations for KM in software development organisations and how SPI can be achieved by certain KM activities. Although our study focus on software development firms and software process improvements, the results and learning are likely to be applicable to other core business processes where knowledge management around processes is critical.

The paper is organised as follows: the next section presents knowledge management in software development and SPI. The third section presents the research approach and introduces the studies. It is followed by presentations and discussions of the two studies. The final section presents conclusions and discusses future research.

2. KNOWLEDGE MANAGEMENT IN SOFTWARE PROCESS IMPROVEMENT

Software process improvement (SPI) is an approach to systematic and continuous improvement of a software producing organisation's ability to produce and deliver quality software within time and budget constraints. SPI emphasises incremental and cumulative improvements and addresses all aspects of the software process; i.e. the practices of the software professionals, planning and production procedures, documentation, organisation, and management (Humphrey, 1989; Paulk *et al.*, 1993). Common SPI approaches are the

Capability Maturity Model (Paulk *et al.*, 1993), SPICE (Emam *et al.*, 1997; Thomson and Mayhew, 1997) and the Experience Factory (Basili *et al.*, 1992a, 1992b, 1993).

The Capability Maturity Model (CMM) defines five maturity levels that lay the foundation for continuous process improvement. Each level contains a set of process goals with the aim to improve an important component of the software process. Except for Level 1, each maturity level is decomposed into several key process areas that indicate the areas an organisation should focus on to improve its software process. Key process areas identify the issues that must be addressed to achieve a maturity level. Software configuration management is e.g. a key process area for maturity level 2.

SPICE (Software Process Improvement and Capability dEtermination) is the name used to describe a set of international standards on software process assessment. SPICE is also the project with the aim to deliver an ISO standard on software process assessment. Similarly to CMM the SPICE model is divided into six capability levels and 5 process areas. The capability levels range from 0 to 5 where level 0 indicates no identifiable good software practices. The process areas cover areas such as engineering, project management etc (Emam *et al.*, 1997; Thomson and Mayhew, 1997). However, capability maturity models have been criticised for being inflexible and making a number of unrealistic assumptions about software development processes (Nielsen and Nörbjerg, 2001).

The Experience Factory is an experiential learning SPI approach (Basili *et al.*, 1992a). It deals with the reuse of software development knowledge and experience (Figure 1). The Experience Factory can either be a logical and/or physical organisation, but it should be separated from the project organisation (Basili *et al.*, 1992b). The Experience Base (a knowledge repository) is a critical component of the Experience Factory. The idea with the Experience Factory is to take the products from the software projects, like plans and data (e.g. error measures) gathered during development and to transform these into reusable units. The units are stored in the Experience Base. From a KM perspective the Experience Factory is primarily based on a knowledge codification strategy and focuses on explicit knowledge.

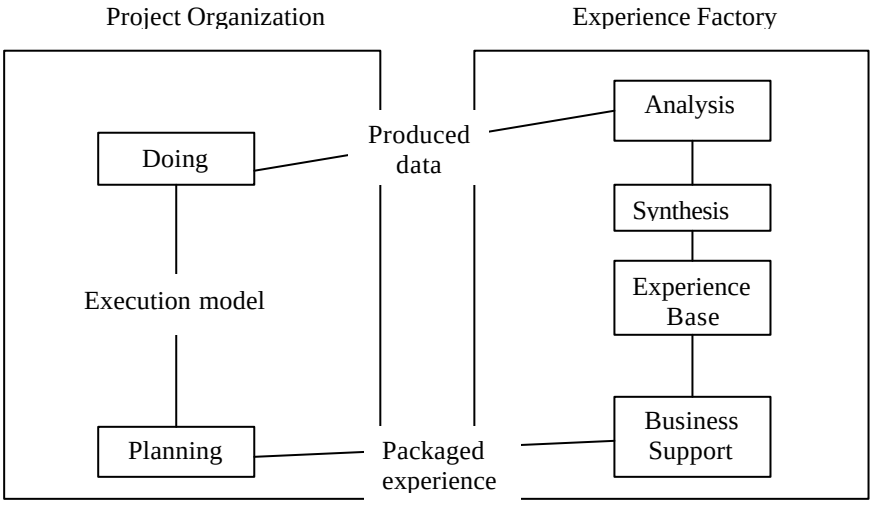

Figure 1. The experience factory model (adapted from Basili *et al.*, 1992b)

Normative models and methods in the software development process have been criticised and their relevance has been questioned (Avison and Fitzgerald, 1999; Truex *et al.*, 2000; Nielsen and Nörbjerg, 2001). Despite the critique of normative models and methods as well as their focus on explicit knowledge, common points of reference and common mechanisms must exist to enable communication and co-operation within and between software projects. In software development both tacit and explicit knowledge are important (Rossi *et al.*, 2000). Methods can play an important role as a common platform for supporting communication and co-ordination. In the SPI literature it has been argued that implementation of methods is one way of improving an organisation's software process (Cugola and Ghezzi, 1998). However, methods have been criticised for increasing development time due to too much emphasis on document production; specifying more steps than are needed in most projects; inflexibility and having a one-size-fits-all view (Avison and Fitzgerald, 1999). Cugola and Ghezzi (1998) found that it has been suggested that software development methods are created to provide expert guidance and wisdom to development processes. Methods also facilitate change and transfer of staff from project to project without retraining. Use of methods is believed to facilitate reuse of knowledge and experiences (Avison and Fitzgerald, 1999). Said Hirschheim *et al.*: "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…" (Hirschheim *et al.*, 1996). Methods can therefore be seen as a mechanism that enables knowledge creation and sharing—knowledge enablers according to Ichijo *et al.* (1998). Knowledge enablers are "…organisational mechanisms for intentionally and consistently developing knowledge in organisations." (ibid.). Ichijo *et al.* (1998) specify three different roles for knowledge enablers: 1) knowledge enablers should *stimulate* individual knowledge development, 2) knowledge enablers should *protect* knowledge development in organisations, and 3) knowledge enablers should *facilitate* the sharing of individual knowledge and experience among organisational members so that individual knowledge will be transformed into organisational knowledge. The use of knowledge enablers in SPI is natural as SPI is a particular form of knowledge creation, sharing and management (Mathiassen and Pourkomeylian, 2001).

Our two studies will focus on the relationship between KM and SPI. Specifically we study one common SPI approach, Experience Factory, from a KM perspective and methods as knowledge enablers.

3. RESEARCH APPROACH AND CASE COMPANIES

The two studies were selected based on relevance to our research area. The first study, in a company called Gamma, is based on interviews, internal documentation, and secondary data (Hellström *et al.*, 1999). The second study, in companies called Theta and Omega, is based on interviews, observations, and internal method related documentation from these two software development companies. Our research was organised as a multiple case study (Yin, 1994). In the Theta case we also drew on personal experiences from methods and tools development and the use of these methods and tools in projects. The documents studied were method descriptions and project specifications for projects using the methods. In the second study we interviewed 15 persons. The interviewees had roles such as programmers, project managers, systems architects, project planners, configuration managers, and in one of the organisations

we also interviewed an IS/IT executive. Ten in depth interviews were done at Omega and five at Theta—one of the authors is an employee at Theta and works with software development. The interviews were done during April and May 2002, and the interviews were face-to-face and semi-structured. Each interview, ranging between 45–90 minutes, were tape-recorded and transcribed. The interviews focused on the use of methods in software projects, what problems the projects faced, how the problems were addressed and solved.

3.1 The case companies

Gamma is a firm within a large telecommunication group (Epsilon). Gamma designs and develops software for management systems of mobile phone networks and telephony and data communication switches. It is located at several places in Sweden. It is organised in business centres and each business centre consists of several business units. A business unit is working with one to three products and has about 20-30 employees. The Gamma case focuses on the firm's attempt to implement the Experience Factory.

The second study was a "snap-shot" study focusing on methods as knowledge enablers. It was carried out in two software development firms: one telecommunication firm (Theta) running projects with over 1,000 people and one smaller firm (Omega) developing military applications with a normal project size of 100 people.

Omega develops, manufactures, and maintains military command and control systems and combat management systems as well as commercial systems. The company has about 750 employees located in four sites in Sweden and abroad. In Sweden Omega consists of five divisions. The division in this study was developing naval systems.

Theta is a company within Epsilon. Epsilon has about 64,000 employees in 120 countries and Theta has around 20,000 employees in various locations in Sweden. Epsilon develops, sells, and maintains telecom systems to operators all over the world. The projects in this study were working with software development for the next generation (3G) mobile phone networks.

4. STUDY GAMMA

A couple of years ago (1997) Gamma decided to start a KM-project with the goal to design and develop a knowledge management system (KMS). The purpose of the KMS was to support in a more formalised way the capturing and re-use of software experiences. It was decided that the KMS should be based on the Experience Factory (Basil, 1993; Basil *et al.*, 1992).

The purpose was to design and implement a computer-based knowledge repository aiming at the collective knowledge of Gamma's software engineers. Gamma's Experience Factory project should develop an Experience Base which should be used to collect, analyse, generalise, formalise, and packet experiences from Gamma's software development projects. (Henceforth, Gamma's knowledge management project is called Knowit and the Gamma Experience Base is called Expbase.) Hence, the project started with a traditional codification strategy to knowledge management (Alavi and Leidner, 2001; Hansen *et al.*, 1999).

Knowit spent more than 1,000 person-hours designing Expbase. It should contain various software metrics related to productivity, like software correctness and fault, lead-time data,

etc. After the initial design, Expbase was presented to its intended users. Based on the reactions of the intended users it was decided that the project should be halted. It was felt that Expbase would not solve the problems encountered in Gamma's software processes and that it would not be a good means to improve software processes.

In parallel with the design and development of Expbase, the project decided to increase its understanding of how experiences were really exchanged within and between Gamma's software projects. The Knowit-manager was supervising a student-thesis focusing on Gamma's software development. The student tracked and described how knowledge was exchanged and distributed as well as what types of knowledge software project members really needed and used. Software project leaders and members were followed over a number of working days. In parallel with the thesis-study, members of Knowit conducted a number of interviews with software project leaders and members. The interviews were focusing on how knowledge was sought and exchanged within and between projects. The results from the two studies showed how experiences were exchanged within projects and between projects. The studies showed major discrepancies between how people were seeking knowledge and exchanging experiences and what Expbase would require people to do to seek knowledge and exchange experiences. Although Gamma had invested more than 1,000 person-hours in designing Expbase, it decided, based on the two studies, to stop the development of Expbase.

Based on studies of how the firm's software developers and project leaders shared and exchanged knowledge in projects and between projects the KM-initiative was re-directed to a people-centred approach (a networking approach). It was decided that an Experience Engine should be designed and that the primary "engine" for knowledge exchange should be humans. Hence, the Experience Engine can be characterised as a personalization or networking strategy to KM (Alavi and Leidner 2001; Hansen *et al.*, 1999). The KM-project studied, for example, over 100 knowledge exchange occasions. The firm's study indicated that to improve knowledge exchange, informal knowledge exchange processes have to be supported and strengthen by formal roles. At the core of the re-directed KM-initiative are two centrally created formal roles: *experience broker* and *experience communicator.* A primary goal of the experience broker is to be a human "yellow pages," and can be characterised as a human networking strategy. The role is focusing lateral relations. The experience broker must have broad and "holistic" knowledge and also have a well-developed network. The experience broker should spend most of his time wandering around in the organisation, moving between different teams, meetings, and joining ad hoc meetings. The experience communicator transfers knowledge and experiences and must be able to present and communicate the experiences in such a way that it helps a software developer in solving a specific problem. The idea is that experience communicators should transfer knowledge that will help a software developer solve a problem in such a way that the software developer increases his knowledge and will be able to deploy this knowledge in further projects. The new and uncommon approach can be characterised as a "knowledge management by wandering and hanging around".

4.1 Study Gamma – discussion

The two project studies made by Gamma suggested that Knowit and the Expbase missed two important dimensions. First, the lack of physical spaces for knowledge exchanges. As noted in the literature, in many cases it is not enough to have virtual spaces for knowledge

exchanges—spatial design can be as crucial in improving knowledge exchanges (Earl, 2001; Brown and Duguid, 2000). Second, Expbase did not seem to be a proper way for handling spontaneous and ad hoc needs for knowledge. For example, the two studies found that during meetings knowledge needs were strongly related to how different issues emerged. Expbase would in these cases not be a suitable support.

The studies also showed that spontaneous and informal knowledge exchanges were frequently used for knowledge exchange. The exchange in these situations was directed towards specific problems that had to be solved "right away". Spontaneous and informal knowledge exchanges were of two types. The first type was networking where a reference was made to a source of knowledge (e.g., a project, person, or a document). The second type involved actual exchange of substantive knowledge. Expbase could have been useful in these situations, but it was assessed that it was unlikely that it should be used. It was likely that the intended users would find Expbase too cumbersome to use compared to face-to-face exchange—the latter was also pointed out as the preferred exchange mode for spontaneous and informal knowledge exchanges.

Tentative lessons from the study are related to: 1) "off-the-shelf" knowledge management and knowledge management systems as a "silver bullet," 2) striking a balance between explorative and exploitative knowledge management, and 3) strategic knowledge management. First, the case illustrates what we call the "silver bullet" approach. The firm looked for a "simple" solution to its problems and also a solution that could be easily communicated. The software developers and project leaders easily understood what the Experience Base was, but the project did not try to develop the Experience Base within an experiential learning framework. They took the Experience Base out of its context. Second, the study illustrates the importance of striking a balance between exploitation and exploration in knowledge management. We suggest that from an organisational point of view, knowledge management as exploitation of the current way to manage knowledge might in the long run lead to some unwanted consequences. The experience broker and experience communicator approach is primarily an anchoring and adjustment approach. This means that the "new" way to manage knowledge and experiences is the result of minor adjustments of the current way to manage knowledge (the anchor point). There was no serious discussion in the project related to the exploration-exploitation issue. Third, in the KM-project there was no strategic view on knowledge management. The project did not try to link knowledge management to the firm's strategic vision and it did not prioritise what to focus and how to manage knowledge in relation to the strategic vision.

5. STUDY THETA AND OMEGA

The second study started with the "assumption" that software development methods can be fruitfully approached from a knowledge enablers perspective—using methods is also a common SPI approach (Cugola and Ghezzi, 1998). The study is a "snap-shot" study of two software development firms and tries to understand what role methods play as knowledge enablers and how they can improve a software process.

In the studied companies various methods related to project management, configuration management, and management of the software engineering process were used. The methods were usually based on international standards from ISO, IEEE, MIL and TickIT. How well

these methods were implemented in the organisations differed significantly. Omega, which was a much smaller organisation than Theta, had a specific set of methods that were commonly known, even if the knowledge about the content of the methods varied from person to person. Theta had a larger number of methods and the method awareness varied largely between units within the organisation. In both organisations, the knowledge and the awareness of the corporate methods varied depending on the level of experience among the project members. Individuals that had a long working experience with the companies knew less about the methods or had difficulties in expressing what the methods actually specified and prescribed. These individuals had more or less "internalised" and personalised the methods. Less experienced employees had a better awareness of the methods and their content. They had fairly recently studied the methods, either by personal interest to increase their understanding of how the organisation worked or in introduction courses. The methods played an important role for the new employees since they needed a "framework" for making sense of their new environment. The method provided them with a point of reference and some basic terminology, which made it easier to spot lack of knowledge as well as it facilitated communication with more experienced colleagues.

In the companies studied it was common to customise the corporate methods for each project. The level of customisation varied. Some projects made significant customisations, e.g. specifying how a method should be used in co-operation with a tool. Others just set up a project web site with methods pages pointing to the corporate method and made only brief comments with regard to the project.

From an SPI perspective it was interesting to note that the feedback from practice to method developers regarding the use and customisation of the methods was remarkably low. Very few interviewees knew how to report suggestions for improvements and customisation experiences. This limited the possibility for process improvements. Reasons for the limited communication between practice and method developers were often related to limited trust between practice and method developers, and distance—the latter both geographical and "mental" distance.

The methods in the study had usually a terminology section, which was an important feature from a knowledge enabling perspective. The terminology were either in the form of a glossary or a special section were key concepts were explained and discussed. The "features" in the methods that focused on the use of a common language showed to be very useful. The glossaries were of extra value in the large Theta projects that had several sub-projects geographically dispersed. The terminology ensured the least common denominator for communication thus facilitating communication and knowledge sharing over national and cultural borders.

5.1 Study Theta and Omega – discussion

Our study suggests that software development methods play an important role as knowledge enablers in the software development process. Ichijo *et al.* (1998) have emphasised the importance for companies to focus on the development of a *common language*, which facilitate understanding and knowledge sharing. If a software development organisation does not have an implemented common method, feedback from projects to method responsible will be reduced, thus complicating the capturing of experiences. The study shows that methods are valuable to new employees. With the help of methods they understand more rapidly what

knowledge they lack, and can catch up faster with experienced people on "…how things are done around here." Methods that specify the minimal level of commonality are essential in this sense since they make collaboration possible, but leave enough room for software teams to solve innovation problems (Cugola and Ghezzi, 1998). Flexible methods are a prerequisite in the fast moving environment of software development and customisations of existing methods were the norm in the case companies. Each development project is a unique undertaking and therefore methods need to be customised to support the specific task (Fitzgerald, 1998). Environmental changes can sometimes be fundamental that minor adjustments of the methods are not effective. When major changes occur new methods need to be developed that incorporates new terminology and workflows that are based on new assumptions about the project's environment. This was evident at Theta, where the major technology shift demanded new ways of working and thus new methods. We argue that product and project complexity and environmental turmoil emphasise the importance for knowledge management and the use of common methods as knowledge enablers in the software development process.

Knowledge management is not only to capture new knowledge and to re-use it in new contexts. To take away obstacles for communication and to enable knowledge creation is just as important (Nonaka *et al.*, 1995; Ichijo *et al.*, 1998; Sheremata, 2001). Based on the study we argue that methods play an important role in reducing communication obstacles. We suggest that an increased exchange between method organisation and projects is important to a software development organisation that wishes to improve its software practice. Experiences in using the method should be fed back to the method development organisation. By having a feedback loop between these two organisational units, knowledge re-use and learning will be improved. Experiences from the project should be taken care of and used to update the model to better fit the requirements of practice and to enrich it with useful insights. If a method is not updated there is risk that its value to practice will diminish, and local customisation efforts from practice will increase. This can lead to a situation where the company ends up with a flora of methods, which is difficult to oversee. As a consequence unnecessary energy is spent on local method development. From a knowledge enabling perspective this is also unfortunate as a common method facilitates communication *between* projects, which is important in a multi-project environment. What can be learned from the cases is that methods in software development are important as knowledge enablers and that they play a central role in an SPI effort.

6. CONCLUSIONS AND FURTHER RESEARCH

The first study concerned the implementation of a KM-solution in a software development organisation, Gamma. The project started with a codification strategy based on the Experience Factory. The project later changed strategy and implemented a network-based KM-solution instead. From an SPI perspective the second approach took the learning and knowledge sharing aspects of process improvement into better consideration. Gamma realised that a networking strategy fitted better under the existing circumstances. Software development and SPI is knowledge intensive work that depends on learning, knowledge creation and sharing. At a first glance the Experience Factory should be a suitable mechanism in this regard as it aims at supporting dissemination and reuse of project related experiences. The problem though is that it relies on statistical data and explicit knowledge that has been taken out of its context. The use of software metrics and codified knowledge is by no means unproblematic

since every software project is unrepeatable in practice. Differences in team skills, learning curve, rapid technical change and changes regarding methods and tools make it difficult to measure one project against another, and to re-use codified knowledge between projects (Boehm and Basili, 2000).

The second study concerned methods as knowledge enablers in software development. The studied projects did not implement methods during our study; neither did they actively pursue knowledge management. The study found however that methods function as knowledge enablers in software development projects. They provide structure to the development process and facilitate the creation of shared mental models and thus support communication within and between projects. They further create a structure for capturing process-related experiences and with structured feedback loops they can play a central role in creating a learning software organisation.

In a software development context that is characterised by a complex project hierarchy of projects and sub-projects, we found that a common method play all the knowledge enabling roles mentioned by Ichijo *et al.* (1998).

1. Methods *stimulate individual knowledge development.* New employees can use methods to identify what they need to know more about and they facilitate conversations with more experienced developers.

2. Methods *protect and support knowledge development.* They take away obstacles for communication and provide the organisation with a communicative framework and contribute to a non-random and unsystematic knowledge development.

3. Methods *facilitate the sharing of individual knowledge and its transformation to organisational knowledge* since they contain common terminology and workflows, thus creating a common ground for communication.

We therefore suggest that methods as knowledge enablers play an important role in SPI, and that it is a relevant SPI-strategy for the future. A method based SPI-strategy should focus on methods that are flexible enough to enable effective customisation in a volatile environment but yet gives enough support for communication and common understanding for the software development process. A method should also be used as a mechanism for learning and knowledge creation around the software development process.

The two studies are quite different but they have a common denominator that software development is a knowledge intensive process and that a knowledge management perspective is useful in an SPI effort. SPI is about improving the software development process, i.e. to improve the efficiency of the process from project to project. As discussed in section two normative capability maturity models have played a central role in SPI until recently, but their limitation in practice have reduced their popularity. Nielsen and Nörbjerg (2001) have discussed this issue and suggest that the failure of capability maturity models depend on that they make some assumptions about organisations that may not be in line with actual circumstances. Organisations are social communities driven by political and power issues rather than being rational organisms. Differences in context and project characteristics limit the efficiency for maturity models. In highly changing work such as software development ready packaged solutions or standard approaches have only a limited effect. This was shown

in the first study where the Expbase failed due to lack of support for context and complexity and unforeseen changes. Thus SPI should focus less on knowledge exploitation approaches and focus more on building learning capabilities and to build enabling mechanisms for knowledge creation and sharing.

The paper discussed among others exploration and exploitation of knowledge in software development organisations, and particularly in SPI. It was noted that normative capability maturity models have a weakness in that they have predominance towards knowledge exploitation and that they make some assumptions that are not longer in line with reality. For future research we therefore would find it interesting to focus on how the maturity models could be improved so that they better consider and support context dependent knowledge creation.

REFERENCES

Aaen, I., J. Arent, L Mathiassen and O. Ngwenyama (2001). "A conceptual MAP of software process improvement". *Scandinavian Journal of Information Systems*, 13, 81-101.

Alavi, M. and D. Leidner (2001). "Review: knowledge management and knowledge management systems: conceptual foundations and research issues". *MIS Quarterly*, 25, 1, 107-136.

Avison, D.E. and G. Fitzgerald (1999) "Information systems development", In *Rethinking management information systems*. W.L. Currie and B. Galliers (Eds), Oxford, UK: Oxford University Press.

Basili, V.R., G. Caldiera, F. McGarry, R. Pajersky, G. Page and S. Waligora (1992a). "The software engineering laboratory--an operational software experience factory", In *Proceedings of the 14th International Conference on Software Engineering*, May 11-15, 1992, Melbourne, Australia.

Basili, V.R., G. Caldiera and G. Canone (1992b). "A reference architecture for the component factory". *ACM Transactions on Software Engineering and Methodology*, 1, 1, 53-80.

Basili, V.R (1993). "The experience factory and its relationship to other improvements paradigms", In *Proceedings of the 4th European Software Engineering Conference*, ESEC '93, Springer-Verlag, 68-83.

Boehm, B. and V. Basili (2000). "Gaining intellectual control of software development". *Computer*, May, 27-33.

Brown, J.S. and P. Duguid (2000). *The social life of information*. Boston, MA: Harvard Business School Press.

Cook, S.D.N and J.S. Brown (1999). "Bridging epistemologies: the generative dance between organizational knowledge and organizational knowing". *Organization Science*. 10, 4, 382-400.

Cugola, G. and C. Ghezzi (1998). "Software processes: a retrospective and a path to the future". *Software Process Improvement and Practice*, 4, 101-123.

Davenport, T., S. Jarvenpaa and M. Beers (1996). "Improving knowledge work processes". *Sloan Management Review*, Summer, 53-64.

Dougherty, D. (1992). "A practice-centered model of organizational renewal through product innovation". *Strategic Management Journal*, 13, 77-92.

Earl, M. (2001). "Knowledge management strategies: toward a taxonomy". *Journal of Management Information Systems*, 18, 1, 215-233

Emam, K.E., B. Smtih and P. Fusaro (1997). "Modelling the reliability of SPICE based assessments". *Software Engineering Standards Symposium and Forum, 1997*. Los Alamitos, CA: IEEE Computer Society.

Fitzgerald, B. (1998). "An empirical investigation into the adoption of systems development methodologies". *Information & Management*, 34, 317-328.

Hansen, M.T., N. Nohria and T. Tierney (1999). "What's your strategy for managing knowledge?" *Harvard Business Review*, 77, 2, 106-116

Hellström, T., U. Malmquist and J. Mikaelsson (2000). "Decentralising knowledge: managing knowledge work in a software engineering firm". *Journal of High Technology Management Research, 2, 3.*

Hirschheim, R., K.H. Klein and K. Lyytinen (1996). "Exploring the intellectual structures of information systems development: a social action theoretic analysis". *Accounting, Management & Information Technology*, 6, 1/2, 1-64.

Humphrey, W. (1989). *Managing the software process.* Reading, MA: Addison-Wesley.

Ichijo, K., G. von Krogh and I. Nonaka (1998). "Knowledge enablers", In *Knowing in firms – understanding, managing and measuring knowledge*. G. von Krogh, J. Roos and D. Kleine (Eds), London, UK: Sage.

Iivari, J. and H. Linger (1999). "Knowledge work as collaborative work: a situated activity theory view. " In *Proceedings of the 32nd Hawaii International Conference on System Sciences.*

Leonard-Barton, D. (1995). *Wellsprings of knowledge.* Boston, MA: Harvard Business School Press.

Løwendhal, B. (2000). *Strategic management of professional service firms*, second edition. Copenhagen: Copenhagen Business School Press.

March, J.G. (1991). "Exploration and exploitation in organizational learning". *Organization Science*, 2, 1, 71-87.

Mathiassen, L., and P. Pourkomeylian (2001). "Managing knowledge in a software organization", In *Software Practice Improvement.* P. Pourkomeylian, Doctoral Thesis, Gothenburg University.

McGrath, M.E., M.T. Anthony and A.R. Shapiro (1992). *Product development: Success through product and cycle-time excellence.* London, UK: Butterworth-Heinemann.

Newell S, M. Robertson, H. Scarbrough and J. Swan (2002). *Managing knowledge work*. Basingstoke, Hampshire, UK: Palgrave.

Nielsen, P-A. and J. Nörbjerg (2001). "Assessing software processes". *Scandinavian Journal of Information Systems*, 13, 23-36.

Nonaka, I. and H. Takeuchi (1995). *The knowledge creating company.* Oxford, UK: Oxford University Press.

Orlikowski, W.J. (2002). "Knowing in practice: enacting a collective capability in distributed organizing". *Organization Science*, 13, 3, May-June 2002, 249-273.

Quintas, P. (1996). "Software by design". In *Communication by design – The politics of information and communication yechnologies.* R. Mansell and R. Silverstone (Eds). New York, NY: Oxford University Press..

Paulk, M. C., C. V. Weber, B. Curtis, and M. B. Chrissis (1993). *The capability maturity model: Guidelines for improving the software process.* Reading, MA: Addison-Wesley.

Rainer, A. and T. Hall (2001). "An analysis of some 'core' studies of software improvements". *Software Process Improvement and Practice*, 6, 169-187.

Rossi, M., J-P. Tolvanen, B. Ramesh, K. Lyytinen and J. Kaipala (2000). " Method rationale in method engineering". In *Proceedings of the 33rd Hawaii International Conference on Systems Sciences.*

Sheremata, W.A., (2002). "Finding and solving problems in software new product development". *The Journal of Product Innovation Management*, 19, 144-158.

Thomson, H.E. and P. Mayhew (1997). "Approaches to software process improvement". *Software Process Improvement and Practice*, 3, 3-17.

Yin, R.K., (1994). *Case study research – Design and methods.* Thousand Oaks, CA: Sage.