Knowledge Management A Facilitator for Software Process Improvement

Mitali Chugh
Assistant Professor
TULAS Instt. Of Engg & Mgmt.
Dehradun, India

Neeraj Chugh
Assistant Professor
UPES,Bidholi
Dehradun,India

D.K. Punia
Head, ISM deptt
UPES,Bidholi
Dehradun, India

ABSTRACT
Software development organizations are focusing on improving the process of software development so that the resultant software is of high quality and the development cost is low from the viewpoint of the competitive advantage. Software process improvement (SPI) is a methodical and continuous improvement approach for the software development processes to develop quality software. SPI establishes a relationship between process maturity and product quality. SPI provides improvement through knowledge creation and collaboration. In flourishing SPI effort Knowledge management is an important element. This paper presents and discusses the various ideas and models for SPI and specifies how knowledge management is significant for SPI.

General Terms
Software Process Improvement, Knowledge Management.

Keywords
Knowledge Management, software Process Improvement.

1. INTRODUCTION
Software is an integral part of individual's daily life. All the facilities which we are habitual of like cars, mobile phones, airplanes, games, factories, banks etc. are controlled by software, The controlling software should be of high quality and highly reliable. According to Standish report on software projects "shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates.

Software organizations have been following a systematic approach of Software Process Improvement (SPI) towards improving the capabilities of software organizations so that they can satisfy the customer's requirements within due constraint of time and at a lower cost keeping in consideration the quality of the software.

SPI was originally developed at Software Engineering Institute (SEI) at Carnegie Mellon University and it was introduced by [22]. It is based on the management of SPI activities, approach adapted for SPI initiatives and perspective used for focusing attention on SPI goals [1]. A number of models and standards for SPI have been developed and used by software organizations as capability maturity model, Capability maturity model Integrated (CMM/CMMI), ISO 9000, BOOTSTRAP, Software Process Improvement and Capability Determination (SPICE).

An organization's software process improvement practice is dependent on the knowledge and competencies of its practitioners and managers [4].On the other hand [11] states that SPI practices depends on the implicit, individual knowledge of practitioners in an organization. For amending the software development practices, the organization should enhance the practitioners' theoretical and practical knowledge of its software practices. Knowledge about the new processes should thus be made available to the levels of organization. However, [9] argue that there is less understanding of sharing of SPI knowledge to all the levels of organization. Moreover it is highly complicated to manage knowledge for the software development.

Other studies have emphasized the usefulness of applying the KM to SPI. [11] And [4] have studied SPI with respect to the process of knowledge creation and suggested the need for more sophisticated models of knowledge creation and expansion as compared to the models provided by Nonaka and Takeuchi[4].

It has also been stated that the core idea of SPI is to make knowledge explicit and its collaboration at different levels of organization. [12]

Kautz and Nielsen[9] have studied the correlation of SPI and Knowledge transfer.[5] has stated SPI with the concept of organizational learning and Kautz and Thaysen [28] studied how knowledge, learning and IT support occur in software organizations. These studies revealed that the concepts and challenges related to knowledge creation, categorization and collaboration have important roles in SPI.

2. SP I METHODS AND MODELS
A software process is a group of activities, methods and tools that are used to build a software product. In the definition of a software process the following information should be considered: activities to be accomplished, necessary resources requested and produced artifacts, adopted procedures and the life cycle model used [19]. Today, SPI has become one of the dominant approaches to improve quality and productivity in software engineering [2].

The six basic principles of SPI by W.S. Humphery are :

1) Major changes to software process must start at the top;

2) Ultimately, Everyone has to be involved;

3) Effective changes require a goal and knowledge of current process;

4) change is continuous;

5) Software process changes will not be retained without conscious effort and periodic reinforcement;

6) SPI requires investment. [22].

All the software development projects are unique and there is no standard development process that can be applied to every kind of a project. However, there is a standard process that should be incorporated to any defined processes. There are a number of reasons to define a software process: [23]

1) An organizational software process minimizes problems related to training, revisions and tool support;

2) The experiences acquired in the projects can be incorporated to the standard process, contributing to improvements in all defined processes;

3) 3) Less time and effort are spent in defining projects' processes.

SPI undergoes a lifecycle in software development process which is a constantly changing software processes and it consists of two phases; i) analyzing process phase and ii) changing process phase [21] and this process will be continuously as depicted in figure 1.

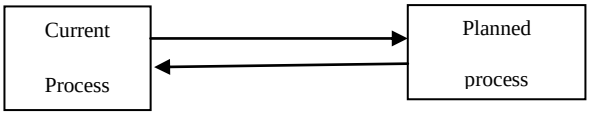

Fig.-1 Change process of SPI

Moreover, there are certain issues in software organizations that can be resolved with SPI: 1) the processes related to how to acquire others knowledge. 2) The processes for knowledge conversion [8].

To summarize, the management of SPI initiatives is based on three ideas: 1) the SPI activities are organized in a dynamic fashion, 2) all improvement efforts are carefully planned, and 3) feedback on effects of software engineering practices is ensured.

The approach to SPI initiatives is guided by three additional ideas: 1) SPI is evolutionary in nature, 2) SPI is based on idealized, normative models of software engineering, and 3) SPI is based on a careful creation and development of commitments between the actors involved. Finally, the perspective in SPI is dominated by three ideas:

1) SPI is focused on software processes,

2) the practitioners' competencies are seen as the key resource, and

3) SPI aims to change the context of the software operation to create sustainable support for the actors involved

2.1 Ideal

IDEAL model comprises five phases (Initiating, Diagnosing, Establishing, Acting, and Learning) provides an unremitting loop through the steps required for software process improvement. In the first or initiating phase the roles, responsibilities and the initial resources for carrying out the SPI project are:

In the diagnosing phase, evaluation activities are designed and executed on the basis of a model to set up a baseline for the organization's current software problems.

The next phase is: establishing, where strategies are developed for carrying out the improvements recognized in the diagnosing phase. Metrics needed to examine progress are defined. In the acting phase, the suggestions for improvement collected in the diagnosing phase are considered and carried out in the organization. Planning is done to pursue pilot projects for testing and evaluating the created software process. The aim of the learning phase is to make the next loop cycle more efficient By this time, the solutions have been developed and documented. Performance Metrics on performance have been collected, and goals have been achieved [13].

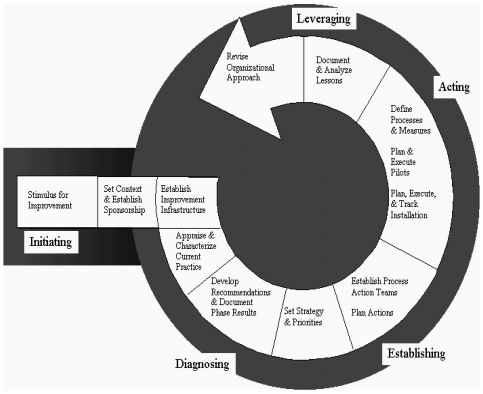

Fig-2 The IDEAL **Model for SPI**

2.2 CMM

CMM is a reference model for inducting the software process maturity into different levels. It describes the process capability of software organizations in five levels. If in an organization the process maturity is higher, the productivity and quality are more and the risk is less.

CMM can be used for capability evaluation and software process assessment. According to CMM software industries can be classified into five maturity levels. Each maturity level in the CMM has several Key Process Areas (KPA) on which an organization at any maturity level needs to concentrate to take it from one maturity level to the next. Each KPA is organized into five sections called common features. The common features specify the best practices that, when addressed together, facilitate to achieve the goals of the key process areas.

CMM has been designed in such a way that it is simple for an organization to gradually build its quality system from scratch.

Level 1: initial this level has no precise requirements and it is characterized by ad hoc activities. It is also known as chaotic level. The software process capability of a level 1

organization cannot be predicted as the software process keeps on changing with the progress of work. Schedules, budgets and product quality are generally unpredictable. Organizations that are on this level have no defined production processes and success in software projects is entirely dependent on individual efforts and heroics. There are no KPAs on this level.

Level 2: repeatable At this level practices for software project management are controlled to track costs, schedules, and functionality. The essential process discipline should be in place to repeat former successes in projects with similar applications and to avoid past failures. The software process capability of organizations at level 2 is characterized as disciplined because the scheduling and tracking of the software project is stable.

Level 3: defined: At this level, documentation is done for both management and development activities because both software engineering and management activities are stable and repeatable. Costs, schedules, and functionality are under control, but the process and product quality is not measured. ISO 9000 aims at achieving this level.

Level 4: managed: At this level the focus is on measurements. Product metrics are used to measure the characteristics of the product being developed(size, reliability, understandability etc.) while the process metrics are used to measure the effectiveness of process being use and to make sure that the software processes operate within statistically predictable limits. The software process capability of the organization can be characterized as predictable because the process is measured and operates within measurable limits.

Level 5: optimized the organization at this level concentrates on continuous process improvement based on quantitative data from level 4. The software process capability of level 5 organizations can be characterized as being in a state of continuous improvement. Improvement is achieved both by incremental advancements in the existing process and by using innovative and new technologies and methods.

2.3 ISO 9000

Another internationally standard approach for SPI is the ISO 9000 approach. The basis of the ISO 9000 standards is that an organization that is disciplined having precisely defined engineering process has higher chance to develop products that constantly meet the requirements of the users, within due constraints of time and available finance, than an undisciplined organization that does not have a well-defined an engineering process. ISO 9001 characterizes 'Quality systems - models for quality assurance in Design/implementation, development, installation and servicing'. A schematic structure of the ISO 9000 standards is given in Fig.3. ISO 9001 is an assembly of quality system requirements that comprises twenty clauses that describes the requirements for quality assurance in design, development, and installation, that defines which aspects of a quality system should be available in an organization. However, the implementation details of these facets are not provided by ISO 9001. ISO 9001 was developed to be functional in all sorts of industries; ISO 9000-3 was further designed specifically for software-development: 'Guidelines for the application of ISO 9001 to the development and maintenance of software'. ISO 9000-3 provides guidelines about the use of ISO 9001 for the specific development that also includes the maintenance of software. The requirements for ISO 9000-3 certification are categorized over twenty requirements classes that are listed in table1.ISO 9000-3 certification guarantees clients that all its processes and work instructions in an organization are documented in such a way that they are strongly related to the ISO requirements, and processes in the certification are being followed on unremitting basis. ISO certification does not provide any assurance about the quality of the product; it only

Table-1 Requirement classes for ISO 90003 Certification

Management responsibilities	Product design requirements
Quality system requirements	Document and data control
Contract review requirements	Purchasing requirements
Customer-supplied products	Corrective and preventive action
Product identification and tracing	Handling, storage and delivery
Process control requirements	Control of quality records
Product inspection and testing	Internal quality audit requirements
Control of inspection equipment	Training requirements
Inspection and test status of products	Servicing requirements
Control of non-conforming products	Statistical techniques

signifies that the measures are utilized up to a particular degree

2.4 Bootstrap

The BOOTSTRAP method is a resultant of a European project under the support of the European Strategic Programme for Research in Information Technology (ESPRIT).

It provides an option for organizations to improve their software development process, and helps in getting ISO 9001 certification. BOOTSTRAP combines the methods supplied by the CMM and the ISO 9000 quality standards. The basis of the BOOTSTRAP method is CMM. Like CMM, it is also based on five maturity levels; however, this method provides different decisive factors for the evaluation of total strength and flaws of an organization.

BOOTSTRAP directly comprises the ISO 9000 quality standards (ISO 9001 and ISO 9000-3) as they provide the state of organization-wide quality system. CMM does not include such guidelines for quality.

BOOTSTRAP mainly identifies three fields that determine the process maturity of an organization: technology, methodology and organization as shown in the fig 4.

Methodology is further categorized as life-cycle dependent, life-cycle independent process areas. A number of processes are defined in the BOOTSTRAP tree. Every process focuses on a number of 'key-practices' that are to be considered for that process.

Moreover, each process has a 'capability dimension', which rates the existing status of the process on a scale from 0 to 5.

Unlike the CMM, between these levels are distinguished sub-levels, which make it possible to evaluate an organization as -Example level 3.5, expressing that level 3 is established and that 50% of the level 4 capabilities are being practiced.

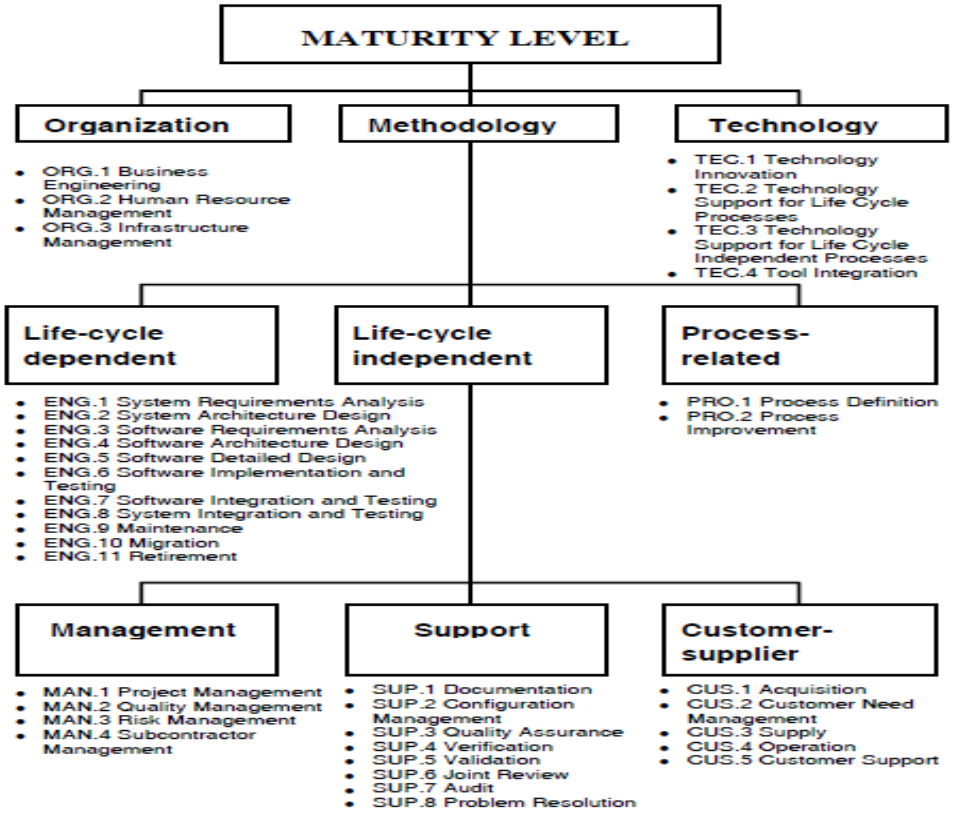

Fig 4: BOOTSTRAP MODEL

2.5 SPICE

SPICE (Software Process Improvement & Capability determination) is one of the widely accepted massive International action for Software Process Assessment [ISO 15504 1998]. ISO 15504 (Figure 5) is used as a reference structure for software process capability determination. It is based on other approaches, as BOOTSTRAP, CMM and ISO 9001.

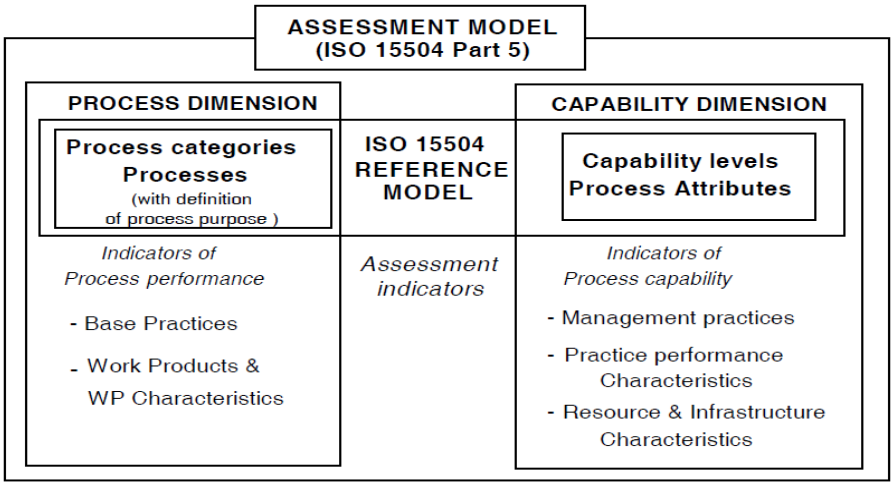

Fig 5 The ISO 15504 framework

Similar to CMM, SPICE is divided into six capability levels and 5 process areas. The levels range from 0 to 5 specifying the extent of software improvement practices. Level 0 indicates that there is no distinct set of software practices. The process areas in SPICE include engineering, project management etc [26]. However, capability maturity models have been condemned for not being flexible and having a number of impracticable assumptions about software development processes [27].

SPICE can be differentiated from CMM in the following ways: 1) ***Scope:*** processes directly associated to the software development processes are also considered. 2) ***Architecture:*** levels are differentiated in all Key Process Areas, whereas in CMM key process areas are significant in a specific level.

3) ***Structure:*** Integration of other SPI-models, such as ISO 9000, Tick IT and Trillium.

3. KM IN SPI

Knowledge is vital resource of an organization and it has to be organized very systematically and carefully. In the early days, organizational knowledge was stored on paper or in minds of the people. However, paper has limited accessibility and updating is a difficult process [7]. Knowledge residing in the mind of people is lost when individuals leave the organization. In addition, in a large organization, it can be cumbersome to localize who knows what. Therefore, knowledge has to be captured, stored, and collaborated systematically across the organization.

Knowledge Management (KM) is basically the management of knowledge resources for creating, retrieving and reusability of knowledge with the help of sophisticated technology. The activities of knowledge management include: creation, capturing, refining, storing, integrating, disseminating, using, and maintaining knowledge.

Knowledge Management is a huge interdisciplinary approach and it has also received attention to a large extent in various fields. There are two main strategies for knowledge management:

Codification – to organize, classify and store information constituting the knowledge of the organization, and to make this accessible to the people in the organization.

Personalization – to maintain the information flow in an organization by having a central repository of information about sources of knowledge, like a ‘‘directory” specifying who knows what in an organization [24].

Knowledge management has been further categorized into schools as ‘‘technocratic”, ‘‘economic” and ‘‘behavioral” [25].

The technocratic schools are: (1) the systems school, which emphasize on technology for sharing of knowledge with the help of knowledge repositories; (2) the cartographic school mainly concentrates on knowledge plans and charts and for creating knowledge directories; and (3) the engineering school, specify the knowledge flows in organizations. The economic school relates to how knowledge possessions are linked to revenues and profits in organizations.

The behavioral school is further classified into three sub schools: (1) the organizational school, states about the interconnections in an organization for sharing of knowledge; (2) The spatial school focuses on how workplace can be designed to facilitate knowledge sharing;

and (3) the strategic school, specifies how knowledge can be viewed as the essence of a organization’s tactics [27].

“The innermost layer of a KM system is organizational memory, facilitating reuse and collaborating organizational knowledge, including lessons learned” [3]. One of the key factors in knowledge management is Information technology [20]. A wide range of technologies are being used to implement KM systems, what includes databases and data warehouses, intranets and internet, browsers and search engines, intelligent agents and so on [7].

In software development knowledge reuse and collaboration are essentials for continuous improvement of the software process and products. As a result of communication between projects and corporate memory two feedback loops are set up.

Feedback loop1: Setup during execution of process, the knowledge attained during the course of the project is analyzed and trivial changes to the execution of the process are applied (process learning).

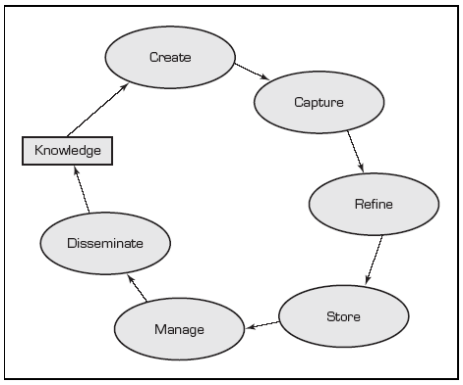

Fig: 6 The Knowledge Management Life Cycle

Feedback loop2: is setup at the end of the project where knowledge is encapsulated, and is used in a new project (corporate learning). [15].

The type of knowledge to be captured in the organization is dependent on the organizational goals and can be used to improve quality. Efficient knowledge storage is also important for knowledge retrieval and reuse.

4. KM CHALLENGES IN SPI

The quality of the knowledge created is dependent on the quality of practitioners' knowledge about the software practices in the organization and the success of the conversion from practical knowledge to process knowledge [4] [5] [11] [14]. One of the major challenges is that of Organizing, planning, and gathering people for SPI-workshops. It is a time-consuming process. It is critical to confine practitioners' knowledge related to SPI practices within a focused area (e.g. software change control); Moreover, it is also a challenging task to convert such insights in the form of explicit knowledge (also known as process knowledge). Another challenge is encountered in implementation efforts, where the explicitly documented knowledge is used by practitioners in practice. There is a deviation from what is expressed in a guideline and implementation practices. When modification is carried out new knowledge is generated which is used in the specific projects. So we can say it is an iterative process. [4][5][16][17].

An additional factor in SPI is knowledge sharing or collaboration. The practitioners at different organizational levels should share different types of knowledge [9] [11] [18]. Practitioner's commitment and acceptance problems arise if sharing activities are not well planned. In addition to this, to compile the results of SPI effort may also be a cumbersome activity without knowledge collaboration. To improve practice, software organizations must learn about SPI [9]. A number of studies disagree that SPI routines and procedures are formal (explicit) procedures, they learnt and developed in practice [6] [12].

According to exploration strategy [5] SPI is supported by sharing knowledge and reflection-in-action. It focuses on changing practice by the creation of personalized knowledge. Actors learn the innovative ways of working through observation or replication and the initiation of strategy is done at the project level. It requires some mechanisms to collect and share experiences, to identify better practices, and subsequently to transfer this knowledge across the organization.

5. CONCLUSION

In this study we have addressed the following research questions. (1) ***What are the major concepts that have been investigated for software Process improvement? (2) What are the major findings on knowledge management in software process improvement? (3) What are the major challenges of Knowledge management in SPI?***

It provides an outline of concerns, ideas and knowledge related challenges to SPI. The software development is a complex process due to the numerous amendments in technologies, processes and multicultural arrangements. For such frequently varying process of software development preformatted packaged solutions or standard approaches are helpful only to a certain extent. SPI is about improving the software development process, i.e. to improve the competency of the process from one project to another. A number of SPI methods and models (CMM, IDEAL, and BOOTSTRAP etc.) have been discussed that are used by the software development organizations for Software process improvement. Software development is a knowledge- and people-intensive activity and Knowledge Management can be viewed as a process for reducing cost, raising employee efficiency, and improving quality of product and services. Knowledge management perspective is helpful in an SPI effort.

Small and steady organizations can possibly survive without knowledge management but for organizations that are huge and scattered, whose environment is rapidly varying, managing knowledge possessions is critical for survival and success. Along with the organized knowledge Management it is also essential that practitioners and managers being the component of an organization must realize that they are able to retrieve some meaningful information from it and they are contributors to KM activity.

Finally, to conclude the Knowledge management challenges in SPI considering viewpoints of researchers the following issues are to be considered and tackled:

a) Organizing, scheduling, and gathering people for SPI-related activities is a cumbersome task.

b) Another challenge is in implementation efforts where there is a deviation from what is expressed in a guideline and implementation practices.

c) Practitioner's commitment and acceptance problems arise if sharing activities are not well planned.

REFERENCES

[1] Aaen, I., Arent, J., Mathiassen, L. and Ngwenyama, O. (2001). A Conceptual MAP of Software Process Improvement. Scandinavian Journal of Information Systems, Special Issue on Trends in the Research on Software Process Improvement in Scandinavia, Vol. 13, pp. 123-146.

[2] Aaen, I., Arent, J., Mathiassen, L. and Ngwenyama, O. 2001. A Conceptual MAP of Software Process Improvement. Scandinavian Journal of Information Systems 13: 81-101.

[3] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn and M. Sintek, "Toward Technology for Organizational Memories",IEEE Intelligent systems, vol. , no. , pp. 40-48, May/June 1998.

[4] Arent, J. and Nørbjerg, J. (2000). Organizational Knowledge Creation: A Multiple Case Analysis. Proceedings of Hawaii International Conference on Systems Science.

[5] Arent, J., Pedersen, M. H. and Nørbjerg, J. (2001). Strategies for Organizational Learning in SPI. In: Mathiassen, L., Pries-Heje, J. and Ngwenyama, O. (Eds.): Improving Software Organisations - From Principles to Practice, Addison-Wesley.

[6] Brown, J. S. and Duguid, P. (1991). Organisational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning and Innovation. Organization Science, Vol. 2, No. 1, pp. 4057.

[7] D. E. O'Leary, "Enterprise Knowledge Management", IEEE Computer, vol. 31, no. 3, pp. 54-61, March 1998.

[8] J.P. Wan and J.M. Yang "Knowledge Management in Software Process Improvement," Application Research of Computer, Vol 19, No.5, 2002,pp. 1-3.

[9] Kautz, K. and Nielsen, P. A. (2001). Knowing and Implementing SPI. In: Mathiassen, L., PriesHeje, J. and Ngwenyama, O. (Eds.): Improving Software Organisations - From Principles to Practice, Addison-Wesley.

[10] M. Broomé and P. Runeson, "Technical Requirements for the Implementation of an Experience Base", in Proc. of the 11 Int. Conference on Software Engineering and Knowledge Engineering , SEKE'99, Kaiserslautern, Germany, 1999.

[11] Mathiassen, L. and Pourkomeylian, P. (2001). Knowledge Management in a Software Process from Principles to Practice, Addison-Wesley. Improvement Unit. International Conference on Managing Knowledge: Conversations and Critiques, University of Leicester, England.

[12] Mathiassen, L., Munk-Madsen, A., Nielsen, P. A. and Stage, J. (1997). "Methods" in Reflective System Development. Vol. 2. Aalborg University: Department of Computer Science, pp. 349-365.

[13] McFeeley, B. (1996). IDEAL. A User's Guide for Software Process Improvement. The Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Handbook CMU/SEI-96-HB-001.

[14] Nonaka, I. and Takeuchi, H. (1995). The Knowledge-Creating Company. Oxford University Press.

[15] Pourkomeylian, P. (2001a). Knowledge Creation in Improving a Software Organization. Proceedings from IFIP WG8.6 Fourth Working Conference on Diffusing Software Product and Process Innovations, Banff, Canada.

[16] Pourkomeylian, P. (2001b). Analysing an SPI Project with the MAP Framework. Scandinavian Journal of Information Systems, Special Issue on Trends in the Research on Software Process Improvement in Scandinavia, Vol. 13, pp. 147-165.

[17] Pourkomeylian, P. (2001c). An Approach to Institutionalisation of Software Processes. Proceedings from Tenth International Conference on Information Systems Development, London, England.

[18] Pourkomeylian, P., Hörnell, J. and Söderberg, S. (2001). Knowledge Sharing in a Software Process Improvement Unit. Proceedings from The Second European Conference on Knowledge Management, Bled, Slovenian.

[19] R. A. Falbo, C.S. Menezes, and A.R.C. Rocha, "Using Ontologies to Improve Knowledge Integration in Software Engineering Environments", in Proceedings of SCI'98/ISAS'98, Orlando, USA, July, 1998.

[20] S. Staab, R. Studer, H.P. Schnurr and Y. Sure, "Knowledge Processes and Ontologies", IEEE Inteligent Systems, vol. , no. , pp. 26-34, January/February 2001.

[21] Stelzer, D. and Mellis, W., (1998). Success factors of organizational change in software process improvement, Software Process: Improvement and Practice, 4,(4), 227 – 250.

[22] W. Humphrey, Managing the software Process, Addison-Wesley Professional, Boston, 1989.

[23] W.S. Humphrey, Managing the Software Process. Addison Wesley Publishing, Company, Massachussets, 1990.

[24] M.T. Hansen, N. Nohria, T. Tierney, What is your strategy for managing knowledge?, Harvard Business Review 77 (2) (1999) 106–116.

[25] M. Earl, Knowledge management strategies: towards a taxonomy, Journal of Management Information Systems 18 (1) (2001) 215–233.

[26] 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.

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

[28] Kautz, K. and Thaysen, K. (2001). Knowledge, Learning and IT Support in a Small Software Company. Proceedings of the European Conference on Information Systems, Bled, Slovenia.