2005

A Pragmatic Approach to Knowledge Management in Software Maintenance

Minal Thakkar
Southern Illinois University Carbondale, mthakkar@siu.edu

Radhakishan Rao Vootla
Union Pacific Technologies, rvootla@up.com

Sanjeev Thakur
Enterprise Rent a Car, sanjeev.thakur@erac.com

Neha Thakkar
Union Pacific Technologies, nthakkar@up.com

A Pragmatic Approach to Knowledge Management in Software Maintenance

Minal Thakkar
Southern Illinois University–Carbondale
Carbondale, Illinois, USA
Email: mthakkar@siu.edu

Radhakishan Rao Vootla
Union Pacific Technologies
Omaha, Nebraska, USA
Email: rvootla@up.com

Sanjeev Thakur
Enterprise Rent-A-Car
Saint Louis, Missouri, USA
sanjeev.thakur@erac.com

Neha Thakkar
Union Pacific Technologies
Omaha, Nebraska, USA
Email: nthakkar@up.com

ABSTRACT

Knowledge used by maintenance engineers (MEs) in resolving a software maintenance (SM) activity is crucial for any organization. Given this criticality of knowledge, attention must be paid to the effective management of knowledge. One of the major challenges to this is to be able to capture the knowledge and thought processes used by MEs and disseminate it to other MEs. Our paper provides a pragmatic solution to this problem by proposing an integrated knowledge management framework for SM. This framework includes a Tacit Knowledge Management System which will be a powerful knowledge repository. We also provide incentive structures that can be used to effectively motivate MEs in using this KM tool.

KEYWORDS

Software maintenance, software engineering, knowledge management, knowledge repository.

INTRODUCTION

Practitioners have been maintaining software for as long as they have been designing it (Glass, 1996). As reported by Nosek and Palvia [1990] and as cited by Sneed [1997], maintenance consumes up to 58% of the total software budget. About 40% to 60% of the software maintenance (SM) effort is devoted to understanding the system (Pfleeger, 2001; Pigoski, 1996).

Maintenance results from the necessity of adapting software systems to an ever-changing environment which often means modifying the software that support organizations' business activities (Dias, Anquetil and Oliveira, 2003).

As reported by Seaman [2002], in understanding software systems, MEs desire to have access to experience and knowledge of original developers, requirements writers and other MEs. Such access is often impossible due to various reasons (Seaman, 2002). The challenge in providing infrastructure and support for knowledge sharing lies in the form of automated tools, and in choosing the content to match the needs of the MEs (Seaman, 2002).

In this paper, we propose an integrated knowledge management framework (IKMF) for SM, and a Tacit Knowledge Management System (TKMS), as one of the components of IKMF.

RESEARCH METHOD

According to Glass [1996], in the area of SM knowledge from research contributes a little than knowledge from practice. The solution presented in this paper is influenced by the researchers' experience and practice in the field of SM. They have vast experience developing and maintaining large scale projects at Fortune 500 companies. They are involved in maintaining software containing approximately over 3 million lines of source code in over 100 modules. The software systems are in maintenance for about eight years.

Apart from researchers' personal experiences, a literature review on the previous research work in the field of SM and knowledge management (KM), and a search on Google[1] for a possible availability of integrated software to capture the tacit

[1] Internet search engine

knowledge of MEs was conducted. The literature search was performed using databases such as IEEE Explore, ACM Digital Library, ABI/Inform, EBSCO Academic Search Premier. The keywords used were KM, experience factory, knowledge repository, knowledge flow, software ontology, personnel turnover in conjunction with SM, software engineering, SM process, SM model, and software process improvement. The search on Google was performed using the keywords software version control, software, source code version control, activity tracking system, and knowledge management.

The limitation of this search is that papers explaining the same concept using different terminology may not have resulted in our investigation.

The alternate method to literature search would be to conduct case studies or formal experiments. Since researchers have experience working in large commercial organizations, a formal case study or experiment would add little to the already known problem.

Before beginning to explain the problem of SM that MEs face and the need to implement a KM process, we briefly introduce the terms used in this paper as it pertains to our study and also look at the previous research work done in this area.

TERMINOLOGY

Knowledge

For the purpose of this paper we use the definition of knowledge in the information systems context. As per this definition, knowledge is part of the hierarchy made up of data, information and knowledge. Data are raw facts. Information is data with context and perspective. Knowledge is information with guidance for action based upon insight and experience (ITIL.com, 2004).

Tacit Knowledge

Tacit knowledge is acquired through one's experience, (Koskinen, 2004).

Explicit Knowledge

According to Koskinen [2004], explicit knowledge, unlike tacit knowledge, can be embodied in a code or a language, and, as a consequence, it can be communicated easily. The code may be words, numbers, or symbols like grammatical statements, mathematical expressions, specifications, manuals, and so forth.

Externalization

According to Nonaka, Takeuchi and Umemoto [1996], the conversion from tacit knowledge to explicit knowledge is termed as externalization.

Knowledge Management

KM includes all activities involved with the generation, dissemination and maintenance of knowledge to meet organizational goals (Desouza and Awazu, 2003).

Software Maintenance

According to the IEEE Standards for Software Maintenance [1993], SM is defined as follows:

"The modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment."

LITERATURE REVIEW

There are various researches conducted in the field of KM for Software Engineering (SE), but only few relate to the process of incorporating a KM tool. It has been argued that even though SM and software development are part of SE, the methodology under which they are carried out are distinct (Maletic and Reynols, 1994; Niessink and Vliet, 2000). The area of SM is unique in the sense that SM is fundamentally a reactive activity, thereby, more chaotic than software development (Dias et al., 2003). The following publications were found to be relevant to our research.

An empirical study conducted by Seaman [2002] on information gathering strategies of software maintainers found that several of the respondents wished they had some kind of lessons learned reports to avoid the previously-committed errors. It was also found that improving the quality of maintenance practices could lengthen the useful life of a software system, thus providing a greater return on investment with respect to the system's original development costs (Seaman, 2002). This paper was one of the inspiring work for us.

Though Capability Maturity Model, a software process improvement approach, suggests that knowledge needs to be managed, it does not explicitly state what knowledge needs to be managed and how, when, where or by and for whom. (Rus and Lindvall, 2002).

Dias et al. [2003] extend Kitchenham et al.'s [1999] ontology for SM to identify and organize the various knowledge domains of SM. Their ontology introduces detailed concepts involved in SM. This paper was of great interest to us as it spans all the steps of SM. However, they do not provide the methodology to store and reuse the tacit knowledge of MEs which we accomplish in our paper.

Rodriguez, Martinez, Vizcaino, Favela, and Piattini [2004] propose a multi-agent system architecture to manage knowledge in SM. Their work clearly focuses on knowledge flow and the sources of information in SM. Nevertheless, the research does not address the methodology to capture and store the tacit knowledge of MEs.

Basilli, Caldiera and Rombach [1994] proposed an experience factory that will capture employee experience and store it in a repository. Hansen [1999] proposed storing skill set of employees in a repository and referred to it as personalization. Ulrike Becker-Kornstaedt and Reinert [2002] proposed process model maintenance through systematic experience capture.

Some of the consulting companies use intranets to allow employees to contribute the knowledge acquired on projects such as code efficiency, process efficiency, etc (Dingsory and Royrvik, 2002; Desouza and Awazu, 2003).

All the above approaches focus on some aspect of KM. However, they lack two things that are important to an ME to carry out the maintenance activity. First, they do not include the causal knowledge (know-why), and relational knowledge (know-with) (Alavi and Leidner, 2001) aspect; second, they do not provide an integrated view that spans all the steps of the software maintenance process as defined by Dias et al. [2003]. To address these two issues, we propose an IKMF containing a TKMS component. The TKMS captures the thought process of the ME and addresses the know-why and know-with issue and the IKMF spans the entire maintenance process and will be useful for MEs and managers.

Also, creating a separate system to store knowledge where it cannot be associated with an SM activity can create knowledge silos. As pointed by Kwan and Balasubramanian [2003], information [knowledge] repositories often are unused and become knowledge silos without interacting with potential user's workspace. Our research will solve this issue by providing knowledge in the context of the user's (MEs, project managers) workspace.

THE PROBLEM

SM is a knowledge intensive activity. MEs need knowledge of the application domain, of the organization using the software, of past and present software engineering practices, of different programming languages, programming skills, etc. (Dias et. al. 2003). Increasing outsourcing projects and offshore development, bringing consultants in-house to work on projects, rotating maintenance staff for production support, organizations downsizing and rightsizing to reduce costs (Desouza et. al. 2003), and high attrition in software industry (Burge and Brown, 2001) do not provide the access to knowledge MEs need in accomplishing the maintenance activity assigned.

This is a concern of almost all organizations having software applications in the maintenance phase irrespective of the size of the team maintaining it. In practice MEs do not document the tacit knowledge used in accomplishing a task. It becomes difficult to manage and meet the project schedule if one of the MEs working on a certain part of the application module leaves or is otherwise unavailable. A knowledge vacuum is created due to the loss of a ME and it takes time to fill this vacuum. Unable to meet the deadline affects the schedule of other inter-dependent teams working together to accomplish the task. The problem seems to be at the lowest level in chain and may not be visible at the top level. But when looked at it with a microscopic view, it is huge. We find this issue as a major challenge for the IT departments and the software industry.

PROPOSED SOLUTION TO THE PROBLEM

Knowledge acquisition and management helps resolve the problem. Because KM is multi-disciplinary, its approaches from various disciplines can benefit SM (Rus and Lindvall, 2002). Rus and Lindvall [2002] cited that an organization's large scale, technology-centered KM approach was replaced with a project need-based approach to knowledge collection and delivery. This approach was successful and led to project staff satisfaction (cited in Rus and Lindvall, 2002). Another KM initiative cited by them is in an integrated-circuit assembly and testing organization. In this implementation, once the problem is solved, its history and solution are stored in a knowledge repository for later reuse.

We follow a similar approach towards implementing KM in SM – addressing local needs and problems in a specific context. Additionally, our research addresses the lack of a process framework in implementing KM in SM.

Solution

There is a real need to capture and externalize the knowledge details such as what approach was taken to solve an issue/defect or change request (procedural knowledge), why the particular approach was taken (causal knowledge), and how it impacts the other applications, databases, etc. (relational knowledge) into a knowledge repository. Usually when new members join the organization or team, work assigned to them is to fix issues and defects. New MEs can search the repository without having to wait for anyone to answer their question on "how to", thus improving their working situation.

From our experience we opine that improving a working situation will lead to improved performance, which in turn will lead to meeting the project deadlines. For this purpose we present a simple, straight forward, intuitive TKMS and an integrated KM framework that does not require a very high level of technical expertise to understand and can be followed even by the non-technical management personnel.

Software Maintenance Activities

While we agree with Swanson's [1980] SM categories, from our experiences and in relation to the knowledge requirements for performing the task itself, SM activities can be classified based on the following matrix:

	Defects	Enhancements	Compliance
Presentation Tier			
Business Tier			
Middleware Tier			
Data Tier			
Infrastructure			

Figure 1: Software Maintenance Activities Matrix

Defects are problems in which the application is not working as per the required behavior.

Enhancements can be functional or performance. Functional enhancements add new or modify existing functionality to the application. Typical performance enhancements are to improve response time or better memory or other resource management.

Compliance requests are to meet government or other compliance regulations. For example, HIPAA[2], Patriot Act[3], and Sarbanes Oxley Act of 2002[4], all forced companies to make changes to their software for compliance.

These activities can occur in any of the tiers shown in figure 1.

---

[2] HIPAA is the acronym for the Health Insurance Portability and Accountability Act of 1996, USA

[3] USA Patriot Act of 2001

[4] Sarbanes Oxley Act of 2002, USA

Activity in the context of this paper is defined as a change request that includes defects, issues, enhancements, compliance, or documentation changes. Activities may result in making changes to multiple files and/or software assets. Activity and change request are used interchangeably in this paper.

The Approach

Tacit knowledge can be acquired and externalized by using a TKMS. This system will require the MEs to document the following when resolving the change requests:

1. What approach/decision was taken? For example: It was decided to add a database column instead of calculating the field in the application every time it was needed.
2. Why was that approach/decision taken? For example: It was investigated that adding a database column would be more efficient than calculating the value at every occurrence since the calculated value was used more than 5 times in the application module. Otherwise the overall application execution time would increase due to calculating the value so many times.
3. Who was consulted for the analysis and decision making? For example: A database designer and the ME himself/herself.
4. What was the impact analysis? For example: Due to this change in the database, another module that runs scheduled job to update the data warehouse had to be changed.

Tacit Knowledge Management System (TKMS)

The TKMS can be utilized by a ME when an activity is assigned. We discuss the TKMS by looking at the two scenarios. One is to capture the tacit knowledge and the other is to retrieve it in the form of explicit knowledge.

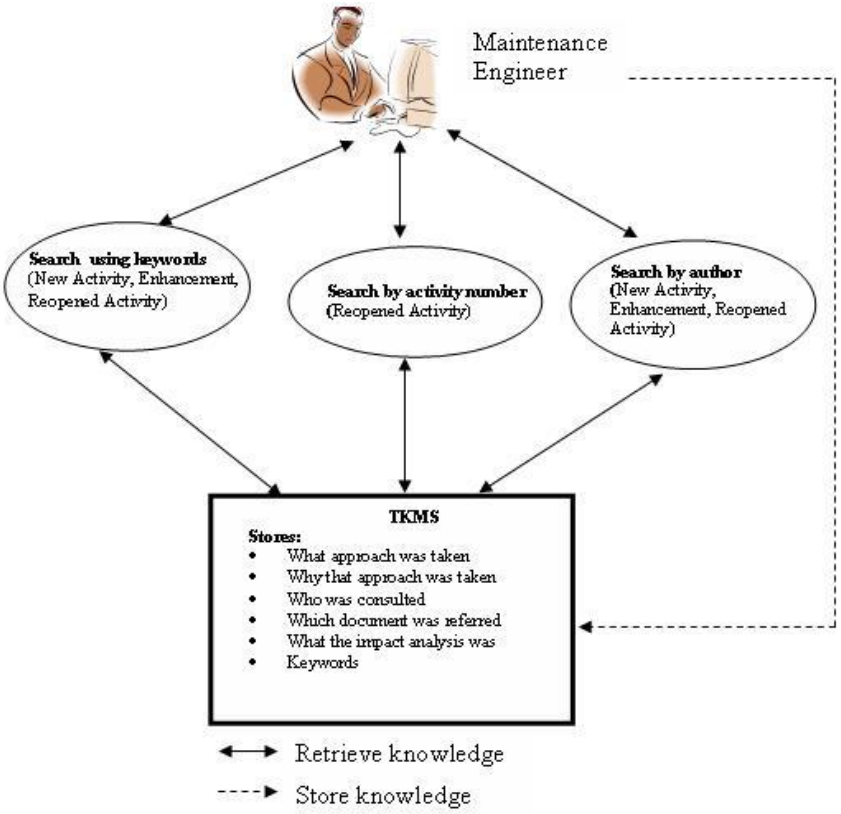

Figure 2: Tacit Knowledge Management System

Scenario 1: Capturing Tacit Knowledge

MEs can use the TKMS while working on the activity or upon completion of the activity. The first step is to identify the type of maintenance based on the matrix in figure 1. It is entirely possible that the request may span more than one tier. Next, the ME working on the activity will store what approach was taken; why it was taken; who was consulted; and the impact

analysis in using that approach along with the activity id or activity description. Keywords relating to that activity and the author's (ME) name are added for easy search in the future.

In order to avoid redundancy, general issues that are not specific to the software system should not be captured and stored in the TKMS. The general rule of thumb should be that if you can "Google" it, it does not belong in the TKMS.

Scenario 2: Retrieving knowledge

Depending on the nature of the change request (see figure 1), the TKMS may be accessed in different ways.

If it is a newly found defect, the ME can either refer to the TKMS for any past knowledge stored on the activity or work on it directly without referring to the TKMS. To refer to the past knowledge stored, the ME can search the TKMS using keywords related to an activity.

If it is a reopened defect, the ME will refer the TKMS by searching through the previous activity id, keywords, or previous ME's name for any possible solutions/approaches/issues/problems. The result is either analysis time is reduced in case a possible solution exists or a different approach is taken in case a certain approach had negative effect on the application previously. The ME can also find a domain specialist (author) through the use of this system based on the keywords for seeking any consultation services.

During the code modification, the ME may conduct an impact analysis to check if the proposed solution has any impact on related modules or dependent applications.

Besides being used as a knowledge repository, another benefit that can be realized from the TKMS is "request mining." Several reports can be generated to highlight trends and frequent problem areas over a period of time. This may help identify areas that need to be focused on in the initial design phase for future projects. For example if most of the requests are in the presentation tier for usability, care should be taken to conduct a thorough usability analysis for future projects.

In order to develop a cost-effective system there are no rigorous technical requirements imposed on implementing the TKMS. While many large companies have implemented KM systems successfully, as reported by Dingsoyr and Conradi (2002) companies are reluctant to invest in an architecture or tool or adoption of a methodology if they do not see any return on investment (ROI). We agree with Rus and Lindvall's (2002) statement that organizations should start KM initiatives on a small scale.

We are currently in the process of implementing TKMS in a Midwestern Fortune 500 company. At the time of this conference, we will be in the design mode of TKMS (Thakkar, Vootla and Thakkar, 2005).

TKMS is a part of an integrated framework (see next section on Framework). We explain TKMS specifically because it is this system that is missing in the current SM component systems. The other SM component systems are being used in almost all organizations. Adding TKMS and interfacing it with other SM component systems makes it a complete and powerful KM system for SM.

Framework

The framework proposed here is generic and can be used by any maintenance team. It identifies the relationship between different SM components and the TKMS to acquire, store and disseminate the tacit knowledge at the point of need.

The framework comprises of five components – Activity Tracking System (ATS), Version Control System (VCS), Configuration Management System (CMS), Quality Assurance (QA), and Tacit Knowledge Management System (TKMS) that directly or indirectly interact with the Program Change Process (PCP).

ATS is an end-to-end activity-based change and defect tracking system. It involves activity management tasks such as creating and managing an activity, assigning it to the appropriate member of the project team, estimating the time required to complete the activity, and communicating with its participants.

For example, when a change request is submitted by the business user, the appropriate project manager is notified [automatically] of the change request. The manager then prioritizes and assigns the activity to the business analyst or a ME along with the deliverable date. They are also notified [automatically] when the manager makes the assignment. The status is updated to "Assigned" in the ATS. The project manager will also be able to track and report the effort put to implement the change.

Once the activity is assigned to a ME, it enters the Program Change Process. Based on the type of the maintenance activity such as defect, enhancement, etc. (see figure 1), the ME may proceed further in one of several different ways (see previous section *Tacit Knowledge Management System*). The tasks accomplished here are - understand the requirement and the code; identify the solution; and conduct impact analysis to identify other modules affected by this change. Once this is done, in a typical SM environment, the ME checks out the files to be changed from a version control system (VCS), makes the necessary changes and unit tests the changes. If the test is successful, the files are checked back into the VCS and the project is "checkpointed" (a version of the files that varies from its previous state). This checkpoint version information is added to the TKMS to be able to locate the actual changes at a later date. A request is then submitted to the CMS for the next step in the process.

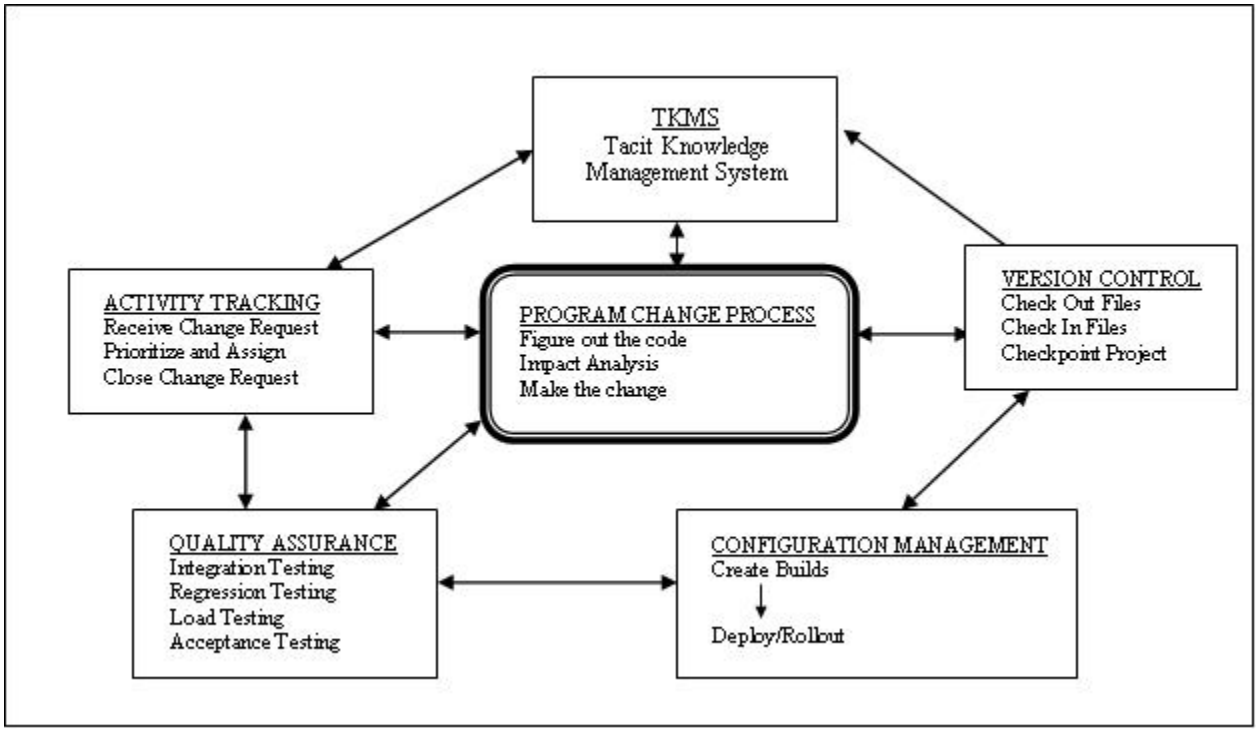

Figure 3: Integrated Knowledge Management Framework for Software Maintenance

The CMS then creates builds for possible deployment and rollout. Before rolling out the application into the production environment, it is sent for QA and testing. The QA team performs the integration testing, regression testing, load testing, and acceptance testing on the change request. Upon successful testing, the QA team will issue a certification to the change request. Once the proper QA certification is received, the change request goes live in production and there after it is closed in the ATS. If the QA team does not certify the change request, it is sent back to the PCP. The whole process in the PCP, CMS and QA is repeated.

This framework provides a single entry point for a software maintenance activity to the users while allowing them to traverse through to other components without having to get out of one component to get to another component. The integrated components result in a system that provides the procedural, causal and relational knowledge in one place.

Incentives

A system is only going to be as good as the self-discipline of its users. Without proper incentives and motivation, knowledge generators inhibit the dissemination of knowledge and use it as a source of power within or against the organization. Even though some companies have incentive programs such as annual bonuses and merit pay increase, these are given based on quantity performance and not quality performance. In other words, employees are not paid based on their contribution of knowledge to the organization. Knowledge providers have less reason to generate knowledge due to lack of sufficient incentives (Desouza et. al. 2003). To properly encourage knowledge contributions, managers and management can make it an employee annual performance measurement and leadership item on which the grade and incentives depend. The more the

employee contributes towards the TKMS, the more points he/she will be awarded on the basis of successful knowledge capture and reuse.

If quality information is stored, there are high chances that a ME would refer to the TKMS first before attempting to resolve an activity. This comes from our years of experience in this field working with other MEs at various companies and is also supported by the findings of Seaman [2002].

We propose some different ways based on which incentives for MEs to build and use the knowledge repository can be calculated.

First, an option should be provided to the ME to rate the knowledge item referred in the TKMS. The rating can be based on the quality, usefulness, relevance, and appropriateness of keywords stored. Ratings from different MEs for each item should be averaged over a period of time to provide incentives.

Second, the quantity of items stored by the ME, and the quality of each of those items should be assessed and proper incentives should be provided.

Third, depending on the number of times each knowledge item stored was accessed. There may be cases were a particular item was never accessed. In that case, the second option suggested above can be used to provide incentives.

Fourth, incentives should be based on whether or not a knowledge item is stored for a completed activity by the ME.

We recommend that managers follow two or more of these incentive structures in combination. Using a single structure by itself may provide undependable results.

CONCLUSION AND FUTURE WORK

The decisions involved in software maintenance are of great significance in terms of the cost and success of the software lifecycle. With little documentation and high attrition rate in software industry, it becomes difficult for MEs to understand the system and the maintenance requirements thus hindering the completion of the maintenance activity. This induces the need to capture the tacit knowledge of MEs.

Previous research has been directed towards what and how knowledge flows in the SM groups. The question on how such knowledge can be captured and externalized for future use is not yet answered for organizations to practically implement it. Our work has taken one step further in principally investigating what knowledge is important to capture in the SM groups as well as proposing an IKMF that encompasses a TKMS and other software components. This proposal can be pragmatically implemented within these groups without creating a financial burden on the organizations.

TKMS will allow a SM team to capture the thought process and knowledge used by the ME in resolving a maintenance activity. When built over a period of time, the KM system will prove to be a great asset to the SM groups.

Implementing the IKMF can lead to efficient and cost effective SM. The framework can track an activity from the request stage to implementation and therefore can assist in system audits.

Currently, our focus is on the process framework of SM and the tacit knowledge storage and retrieval. Thus, we use only the externalization concept of Nonaka [1996]. Other concepts can be included in this framework in the future. The proposed framework will be useful to guide researchers studying knowledge management in other phases of software engineering. Our work will allow researchers in the KM field to focus on procedural, causal, and relational knowledge taxonomies thus creating learning organizations.

This research can also be used as a basis to develop an integrated software system based on IKMF which we plan to pursue in the future. Future research can include research on the maintenance and archiving of knowledge stored in TKMS.

REFERENCES

1. Alavi, M. and Leidner, D. E. (2001) Review: Knowledge management knowledge management systems conceptual foundations and research issues. *MIS Quarterly*, 25, 1, 107-136.
2. Basili, V. R., Caldiera, G. and Rombach, H. D. (1994) The experience factory. In J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering* (469-476). John Wiley.

3. Burge, J. E. and Brown D. C. (2001) Design rationale for software maintenance, *Proceedings of 16th Annual International Conference on Automated Software Engineering*.
4. Desouza, K. C. and Awazu, Y. (2003) Knowledge management, *HR Magazine*, 48, 11, 107-112.
5. Dias, M. G. B., Anquetil, N. and de Oliveira, K. M. (2003) Organizing the knowledge used in software maintenance, *Journal of Universal Computer Science*, 9, 7, 641-658.
6. Dingsoyr, T. & Conradi, R. (2002) A survey of case studies of the use of knowledge management in software engineering, *International Journal of Software Engineering and Knowledge Engineering*, 12, 4, 391-414.
7. Glass, R. L. (1996) The relationship between theory and practice in software engineering, *Communications of the ACM*, 39, 11, 11-13.
8. Hansen, M. T. (1999) The search-transfer problem: The role of weak ties in sharing knowledge across organizational subunits, *Administrative Science Quarterly*, 44, 82-111.
9. IEEE Standard for Software Maintenance (1993) Std. 1219-1993, *IEEE Computer Society*.
10. ITIL.com, Retrieved November 26, 2005 from http://www.itilpeople.com/Glossary/Glossary_k.htm.
11. Kitchnhami, B. A., Travassos, G. H., Mayrhauser, A. V., Niessink, F., Schneidewind, N. F., Singer, J., et al. (1999) Towards an ontology of software maintenance, *Journal of Software Maintenance: Research and Practice*, 11, 365-389.
12. Koskinen, K. U. (2004) Knowledge Management to improve project communication and implementation, *Project Management Journal*, 35, 2, 13-19.
13. Kwan, M. M. and Balasubramanian, P. (2003) KnowledgeScope: Managing knowledge in context, *Decision Support Systems*, 35, 4, 467-486.
14. Maletic, J. I. and Reynolds, R. G. (1994) A tool to support knowledge based software maintenance: the software service bay. *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*,11-17.
15. Niessink, F. and Vliet, H.V. (2000) Software maintenance from a service perspective. *Journal Of Software Maintenance: Research And Practice*, 12, 103–120.
16. Nonaka, I., Takeuchi, H. and Umemoto, K. (1996) A theory of organizational knowledge creation, *International Journal of Technology Management*, 11, 7/8, 833-845.
17. Nosek I & Palvia P (1990) Software maintenance management-changes in the last decade, *Journal of Software Maintenance*, 2, 3, 157-174.
18. Pfleeger S. L. (2001). *Software engineering: Theory and Practice*. (2nd ed.). New-Jersey: Prentice Hall, 475.
19. Pigoski, T. M. (1996) *Practical software maintenance: Best practices for managing your software investment*, John Wiley & Sons, 35.
20. Rodiguez, O. M., Martinez, A. I., Vizcaino, A., Favela, J. and Piattini M. (2004) Identifying knowledge management needs in software maintenance groups: A qualitative approach, *Proceedings of the Fifth Mexican International Conference in Computer Science*, Published by IEEE Computer Society.
21. Rus, I. and Lindvall, M. (2002) Knowledge management in software engineering. *IEEE Software*, 19, 3, 26-38.
22. Seaman, C. B. (2002) The information gathering strategies of software maintainers, *Proceedings of the International Conference on Software Maintenance*, 141-149.
23. Sneed, H. M. (1997) Measuring the performance of a software maintenance department, *Software Maintenance and Engineering, First Euromicro Conference*, 119-127.
24. Swanson, E. B. (1980) *Software maintenance management: A study of the maintenance of computer application software in 487 data processing organizations*. Addison-Wesley, Reading, MA.
25. Thakkar, M., Vootla, R. and Thakkar, N. (2005) Knowledge Management in Software Maintenance: From Theory to Practice (Research in Progress). Proceedings of the Third Annual International Conference on Information Science, Technology and Management (CISTM), July 24-July 26, 2005.
26. Ulrike Becker-Kornstaedt and Reinert R. (2002) A concept to support process model maintenance through systematic experience capture, *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, 465-468.
27. Vliet, H. V. (2000) *Software engineering: Principles and practice*, (2nd ed.). John Wiley Publishers.