KNOWLEDGE DISCOVERY IN SOFTWARE PROCESS IMPROVEMENTS

Ms.Aarti[1], Mrs Anjali NamDev[2]
M.Tech Student, Assistant Professor, Departments of Computer Science and Engineering
MDU University Rohtak (Haryana)

Abstract: ***We present a Knowledge Discovery in Software Process Improvement method which is based on exceeds budget and deliver products with poor quality are abundant in the literature. The role of knowledge components and a knowledge driven model (KDM) are assessed by a measurement model. Insights from the field of knowledge management are therefore potentially used in SPI efforts to facilitate the creation, modification, sharing of software process in any organization. Software Process Improvement setting: Mentoring, RUP, Process Workshops and Post Mortem Analysis. The impact of KDM on the end-product and its real effect on SPI is measured by quantifying the productivity of the projects, eventually the organization. Software is described by its capabilities . The capabilities relate, the features it provides and the facilities it offers. Software written for sales-orders processing would have different functions to process different types of sales order from different market segments. The software is developed keeping in mind certain hardware and operating system considerations known as platform . A major challenge is to create strategies and mechanism for managing relevant and updated knowledge about Software development and maintenance .***

Keywords: SPI, RUP, Mentoring, Workshop and PMA.

I. INTRODUCTION

Software is a set of instructions to acquire inputs and to manipulate them to produce the desired output in terms of functions and performance as determined by the user of the software. Software process improvement is that improving the process will lead to improvements in the final product. SPI is an applied academic field drawing on its roots in both the software engineering and information systems disciplines. The field takes a managerial approach rather than dealing directly with the techniques used to write code, and it deals primarily with

managing software firms to improve their practice. It states that if the general expectation within software engineering is that software will not work properly and a crisis-filled environment are reasonable indications, then software engineering is indeed a profession in a continuing state of crisis.

1.2 Levels of software

- Machine Micro logic
- Supervisor or Executive
- Operating System
- Language Translators
- Utility Programs
- Inquiry, File, and Database Software
- Programming and Assembly Languages and programs
- 4GL Language and User Programs such as SPSS, dbase and SQL, etc.

1.3 Types of Software

There are many different types of software. One of the most important distinction is between Custom software, generic software and embedded software.

Custom software is developed to meet the specific needs of a particular customer and tends to be of little use to others. Much custom software is developed in-house with in the same organization that uses it. Examples of Custom software include web sites, air-traffic control systems and software for managing the specialized finances of large organization.

Generic Software, on the other hand , is designed to be sold on the open market , to perform functions that many people need, and to run on general-purpose computers. Generic software is often Commercial Off-The-Self software (COTS), and it is sometimes also called shrink-wrapped software since it is commonly sold in packages wrapped in plastic. Examples of generic software include word processors, operating systems, computer games and accounting packages for small businesses.

Embedded software runs specific hardware devices, which are typically sold, on the open market. Such devices include washing machines, DVD players.

	Custom	Generic	Embedded
Number of copies in use	Low	Medium	High
Total processing power devoted to running this type of software	Low	High	Medium
Worldwide annual development effort	High	Medium	Medium

Table2.2 summarizes some of the important characteristics of custom, generic and embedded software.

II. LITERATURE REVIEW

How can Knowledge Management be applied to Software Engineering in order to faster Software Process Improvement?

Given our setting where company strategies frequently changed, it proved invaluable to have the overall focus of

looking at software process improvement efforts from the perspective of the knowledge management. In cooperation with the participating companies, we agreed on concrete methods and settings.
Application of the knowledge management to improve the software process through codification of knowledge.
This theme concerns the codification strategy. In order to investigate this strategy, we chose two companies that wanted to improve their software process through different codification of initiatives. As researchers we had some influence in what methods they used to define their process, but in the end it was the companies' decision on what they wanted to spend their time and resources on. The following concrete research questions were answered in the studies which relates to codification. In order to improve codification it was interesting to know what artifacts the developers themselves found useful. This information was considered useful irrespective of which method was used to codify it. One way to improve the software process is to codify it in a process model. A practical framework for codifying such a process that has gained widespread use in industry is the Rational Unified Process. Despite its popularity there was not much published material on the challenges of adapting such a comprehensive framework to a small or medium sized setting.

How can knowledge transfer through a mentor program be improved?
One way to transfer knowledge from person to person is the mentoring approach. We wanted to know how it functioned in the context of a medium sized software company, and if we could improve it using theories from the research field.

How can sharing of project experience through project retrospectives be improved?
A way of transferring experience from person to person is project retrospectives. We proposed changes to the brainstorming in the root cause analysis phase of one such method and wanted to test if this was an improvement on the method and if so, what that improvement consisted of.

2.1 Research Process
The research process for this thesis has been iterative and a lot of projects have happened in parallel, mutually influencing each other. The work can roughly be divided into three main directions:
Three industrial case studies

- Study1: A study of mentoring for transfer of knowledge
- Study2: A study on codifying the software process through an adaptation of RUP
- Study3: A study on reaching an agreement on and codifying the software process through the process workshop
- Study4: A study on using and improving the post mortem analysis to elicit experience from a finished project.
- Study5: A literature study (using systematic review).

Study 1: Mentoring Our reason for looking closer at this company was that they expressed an interest in "improving internal knowledge management through revised work processes and internal training of employees in new processes". Particular for this company, was that they were very interested in the human aspects of knowledge sharing, not just codifying the knowledge.
Study 2: RUP The company utilizes their high competence in RUP and most projects are more or less inspired by RUP, however, the company's management saw a need and a possibility to improve their use of RUP by adapting and codifying their development process to the RUP framework. The company wanted to adapt the rational unified process to their projects. Our first intervention was to help the company define their project types. We then held several workshops trying to adapt RUP from a top down perspective but it was soon evident that we had to rethink this strategy. The company then held a series of smaller workshops where the researchers were just observers, and more people of the company were involved.

Study 3: Process Workshops Their main activities are hiring out consultants as developers, developing complete solutions for customers, and hiring out consultants and project managers as advisors for selecting technology, strategy or process. Typically, no more than four to five consultants are at any time working for the same customer. One of the identified stumbling blocks for experience sharing and reuse was the lack of a common process and a common set of document templates. In order to remove, or at least reduce this problem, the company wanted to define, document and implement a framework that could be used for development, consultancy and operation.
Study 4: Controlled Experiment on PMA
This study began quite informally. In our studies of companies in SPIKE, we sometimes used a method for retrospective analysis, called the post mortem analysis. The method is Research Plan 39 designed to extract important positive and negative experiences from finished projects, or projects at phase transitions and analyze the key causes of these experiences in order to improve future projects.
Study 5: Systematic Literature Review
The major reason for choosing systematic review over a regular literature review was the desire to get results that was replicable and the possibility to assess the completeness of the search. In order to meet deadlines and keep the workload manageable we needed to limit our search somewhat. We were also unable to find any good survey papers that properly covered the field and that was up to date. We therefore decided to make a systematic review on knowledge management in software engineering.

III. IMPLEMENTATION PROBLEMS

The observations given below identify some common barriers to implementation of software process improvement methods experienced by the case study sites, and some of the techniques used to overcome these barriers.

- Getting started: Some of the organizations had difficulty getting started with software process improvement and the methods they selected. We

encouraged these organizations to undergo assessment as a proven technique for helping to identify their priorities and get buy-in across the organization for process improvement activities.

- Staff turnover: Some of the organizations have been involved in downsizing (layoff) activities which affected software engineering staff turnover. We have also observed that within any organization, certain champions and advocates of software process improvement exist. If these individuals are affected or their priorities change as a result of downsizing, then introduction of new software process improvement methods is slower and more difficult.
- Dedicated resources: Some of the organizations utilized part-time resources, usually line managers or improvement teams, to implement software process improvement methods. Although this issue is greatly dependent on the size of the organization and the specific skills and influence of the individuals involved, part-time effort on process improvement is usually not as effective as when full-time dedicated resources are used.

Software Process Improvement is a long term incremental activity. Process improvement involves process analysis, standardization, measurement and change. We do process improvement because we want to build better products (cheaper, more dependable, quicker ...), We really don't know how to measure the product characteristics.

IV. CONCLUSION

- If the PWS approach is used to reach an agreement on the current process, a good starting point is to focus the discussion on artifacts, or what should be produced, rather than how it should be produced.
- If the PWS approach is used to specify future processes based on best practice, the discussions should be focused towards activities, or how the projects should be run.
- The PWS approach is a good tool for organizational learning.
- Involvement in the workshops fosters ownership of the resulting process, and as such it is a good way to get the developers to actually use the process later.
- The process workshop is a lightweight approach to defining a process for companies. As such it is well suited to small and medium sized companies. It does, however, require some resources to be truly successful and therefore, management support is important.

V. FUTURE WORK

We have investigated a mentor program in a small software consulting company in order to identify issues that could be improved. We found many different mentor schemes to be in place in the company, found arguments in favor and against a more formal approach to mentoring in the company. We also made a clearer separation of roles, and suggested that mentoring should have a greater availability in the company. We believe that the new mentoring program will provide better support for double loop learning through increased reflection. The amount of reflection should increase when the mentors pose more open questions during meetings. Also, organizing mentoring in a group of protégés should lead to more discussion, which should also lead to more reflection on current work practices. The new mentoring program has been introduced through a meeting with all employees, and now that the work of restructuring the mentor program is done, we switch to an observer role. We will follow mentor and protégé pairs in new projects and evaluate the changes brought on by redefining the mentor program. The next challenge for the scientists will be to come up with good methods for extracting most of the experience of the employees in a way that is not too intrusive to the regular work of the company, yet still captures the most crucial knowledge.

REFERENCES

[1] Jacobson, I., G. Booch, and J. Rumbaugh, The Unified Software Development Processed. A.W. Longman. 1999, Reading: Addison Wesley Longman. 463.
[2] Krutchen, P., The Rational Unified Process: An Introduction 2nd 2000: Addison-Wesley. 298.
[3] Bergström, S., Råberg, L., Adopting the Rational Unified Process. 2004,
[4] Addison-Wesley. p. 165-182.
[5] http://www.m-w.com/dictionary.htm
[6] Nonaka, I., Takeuchi, H., The Knowledge-Creating Company. 1995:
[7] Avison, D., Action Research. Communications of the ACM, 1999. 42(1): p. 94.
[8] T. Dybå, "Improvisation in Small Software Organizations", IEEE Software.
[9] A.Wickert and R. Herschel, "Knowledge management issues for smaller businesses", Journal of Knowledge Management, no. 4, vol. 5, pp. 329-337, 2001.
[10] M. Lindvall and I. Rus, "Knowledge Management in Software Engineering", IEEE Software, no. 3, vol. 19, pp. 26-38, 2002.
[11] F. J. Armour and M. Gupta, "Mentoring for Success", IEEE IT Pro, no. May - June, pp. 64-66, 1999.
[12] K. E. Kram, Mentoring at work: Developmental relationships in organizational life. Glenview, IL: Scott Foresman, 1985, ISBN: 081916755X.
[13] B. R. Ragins, J. L. Cotton, and J. S. Miller, "Marginal Mentoring: The Effects of Type of Mentor, Quality of Relationship, and Program Design on Work and Career Attitudes", Academy of Management Journal, no. 6, vol. 43, pp. 1177-1194, 2000.
[14] Webster's, Encyclopedic Unabridged Dictionary of the English Language. New York: Gramercy Books, 1989.
[15] F.F. Fajtak, Kick-off Workshops and Project

Retrospectives: A good learning software organization practice, Proceedings of the 7th International Workshop on Learning Software Organizations

[16] J.S. Edwards, Managing Software Engineers and Their Knowledge, in: A. Aurum, et al. (Eds.), Managing Software Engineering Knowledge, Springer- Verlag, Berlin, 2003, pp. 5-27.

