A FRAMEWORK FOR MANAGING SOFTWARE ENGINEERING KNOWLEDGE

Suyeon Kim[1)], Euiho Suh[2)], Hyunseok Hwang[3)]

[1)] Pohang University of Science and Technology, Korea (tomi@postech.ac.kr)
[2)] Pohang University of Science and Technology, Korea (ehsuh@postech.ac.kr)
[3)] Pohang University of Science and Technology, Korea (ieman@postech.ac.kr)

Abstract

Software development is a knowledge-intensive work. The productivity of a software developer and the quality of the software is highly dependent on the experience and skills of the individual. Knowledge therefore plays an extremely important role in software engineering.

In this paper, we propose a framework for managing the software knowledge effectively. We identify the taxonomy of knowledge created during software development life cycle, and then suggest a method for extracting and sharing the knowledge toward Quality Software Factory. The proposed framework is based on software development and consulting experiences in many large-scale projects.

Keywords: knowledge management, software engineering knowledge, knowledge extraction, knowledge sharing

1. Introduction

Knowledge sharing and reuse among software developers is a critical factor for the success of software development projects since the software development life cycle inherently consists of a number of knowledge-intensive tasks. Land (2001) emphasizes the importance of learning from past development experience to ensure we do not make the same mistake repeatedly, and to avoid wasting precious resources by reinventing the wheel.

In this paper, we first propose a QSF (Quality Software Factory) model for implementing quality software. Next, we identify the knowledge created during software development based on the QSF model. We suggest a framework for managing software engineering knowledge toward QSF. In the proposed framework, we present knowledge extraction techniques and knowledge sharing methods based on Nonaka's SECI (Socialization, Externalization, Combination, Internalization) model. The proposed framework is adopted to several large-scale projects in the financial industry. Critical success factors for managing SE knowledge are extracted from our experiences.

2. Quality Software Factory

Software engineering is a problem-solving activity and the chief purpose is to develop quality software [Nunamaker, 1991].

We established the QSF (quality software factory) model for quality software development. QSF is a concept for improving software quality by focusing on the quality of all processes in software development, as well as that of the final software product. QSF can be achieved by providing robust disciplines such as standard, guideline, and quality

assurance functions enforced in all software development activities.

The QSF model (shown in Figure 1) involves the overall methodology of software development life cycle, including phase, project management, development coordination, quality assurance and role.

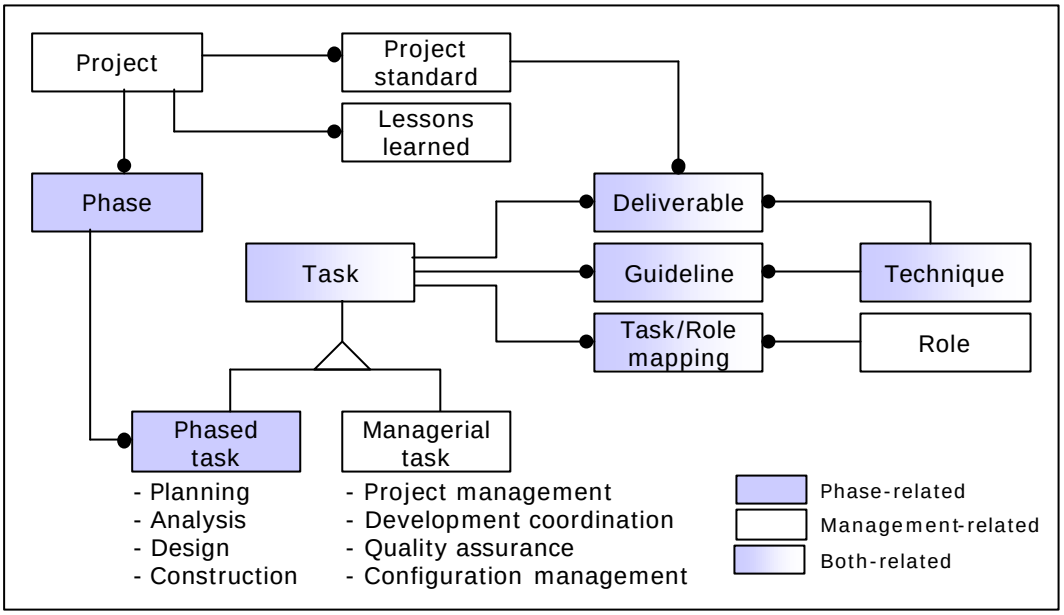

Figure 1. Meta-model of software development methodology for QSF

3. Taxonomy of software engineering knowledge

We identify and categorize software engineering knowledge based on the meta-model represented in Figure 1. Software engineering knowledge can be classified into primary and supporting knowledge. Primary knowledge is relevant to each phase in the software development life cycle: planning, analysis, design, and construction. Supporting knowledge covers all knowledge in software engineering management activities. SE management activities include project management, development coordination, quality assurance, and configuration management. Table 1 represents taxonomy of software engineering knowledge.

Table 1. Taxonomy of SE knowledge

Primary Knowledge (SE Phase-related)	
Planning	Business strategy understanding, Information needs, IT strategy
Analysis	Domain knowledge, Modeling knowledge (e.g., ER modeling)
Design	Design technique (data design, architectural design, procedural design)
Construction	Target programming language, Target platform knowledge
Supporting Knowledge (SE Management-related)	
Project mgmt	Project standard, Role specification, Project scheduling
Dev. coordination	Architecture administration (model, repository), Project coordination
Quality assurance	Software quality factors and metrics, Review guidelines
Configuration mgmt	Software configuration items, SCM process, SCM standards

4. Knowledge extraction and sharing

Software engineering knowledge can be extracted from various sources: software developers, domain experts, methodology experts, existing documents in the company, external information, and project deliverables.

We can extract software experiential data using techniques presented in Table 2.

Table 2. SE knowledge extraction techniques

Primary Knowledge (SE Phase-related)	
Planning	Interviewing for planning, Executive intensive planning, Questionnaire
Analysis	Interviewing for analysis, JRP (Joint Requirements Planning), Questionnaire
Design	JAD (Joint Application Design), Prototyping
Construction	Document assembly and production
Supporting Knowledge (SE Management-related)	
Project mgmt	Project approach selection, Project estimation, Metrification and reporting
Dev. coordination	Model management, Project coordination
Quality assurance	Structured walkthrough, Software inspection
Configuration mgmt	Version control, Change control, Configuration audit, Status reporting

We placed all knowledge created in each phase of the software development life cycle into a central repository. Almost project deliverables, including entity relationship diagram and involvement matrices, are stored in the repository provided by CASE tool automatically.

Supporting knowledge relevant to managerial activities is stored and shared in a web-based collaborative environment. Human contact, such as training, brainstorming, and review meeting, can be also utilized for transferring and sharing knowledge among software developers.

We propose methods for sharing software engineering knowledge based on the Nonaka's SECI model [Nonaka, 1995]. According to the SECI model, knowledge creation processes are categorized into socialization, externalization, combination, and internalization, as shown in Table 3.

Table 3. SE knowledge sharing methods

Socialization (implicit->implicit)	**Externalization** (implicit->explicit)
Apprenticeship	Modeling process
Project experience sharing	Documentation
Training	Learning history [Kleiner, 1998]
Internalization (explicit->implicit)	**Combination** (explicit->explicit)
Document investigation	Digitalization of written material
Guideline understanding	Consolidation of dispersed output
Software skill learning	Executive summary reporting

5. Conclusion

All software development projects have a single essential goal: to produce high-quality software. Even the most jaded software developers will agree that high-quality software is important. To achieve this goal, it is very important to capture and share software knowledge created during the entire software life cycle because software development is highly knowledge- and experience-intensive work.

We have proposed a framework for managing software knowledge based on the QSF model. Critical success factors for QSF extracted from our experiences in the software development can be summarized as follows: making an effective communication channel, transparent progress management, encouragement of knowledge sharing mind, efficiency of knowledge store/access procedures and minimization of manual work. We expect that this framework can be utilized practically and effectively in software development environments.

References

[Boehm, 1981] Barry W. Boehm, *Software Engineering Economics*, Prentice-Hall, Inc., 1981.

[Kleiner, 1998] Kleinder, A., Roth G., How to make experience your company's best teacher, *Harvard Business Review on Knowledge Management*, Harvard Business School Press, 1998.

[Nonaka, 1995] Nonaka Ikujiro, Hirotaka Takeuchi, *The Knowledge-Creating Company*, Oxford University Press, 1995.

[Land, 2001] Lesley Pek Wee Land, Aybüke Aurum, Meliha Handzic, Capturing Implicit Software Engineering Knowledge, Proceedings of Software Engineering Conference, Australia, 108-114, 2001.

[Nunamaker, 1991] Nunamaker, J. F., Dennis, A. R., Valacich, J. S., Vogel, D. R., George, J. F., Electronic meeting systems to support group work, *Communications of the ACM*, 34(7), 40-61, 1991.

[Lee, 1993] Hing Yang Lee, Software Engineering Knowledge for Software Reuse, Proceedings of the Sixth International Workshop on Computer-Aided Software Engineering, 263 –269, 1993.