An Extended Knowledge Management Framework During Software Development Life Cycle

Ali Ahmad Alawneh

Ph.D. Student in MIS, Faculty of Information Systems & Technology, Arab Academy for Banking & Financial Sciences, Amman- Jordan
alawneh2001@yahoo.com

ABSTRACT

Knowledge is one of the organization's most important values that influencing its competitiveness. One way to capture organization's knowledge and make it available to all their members is through the use of knowledge management systems. In this paper I discussed the importance of knowledge management in software development and I presented an infrastructure to deal with knowledge management in software engineering environments (SEEs).
Knowledge is one of the organization's most valuable assets. In the context of software development, knowledge management can be used to capture knowledge and experience generated during the software process

*This Research paper addresses a new way of thinking about the role of knowledge management in software engineering environments through developing a new extended hybrid framework that combines a five types of knowledge(user requirements knowledge, functional domain knowledge, technical knowledge, project status knowledge, and project experience knowledge) with five phases of software development (planning, analysis, design, implementation , and maintenance & support) with five phases of knowledge management life cycle(capture, creation , codification, communication, and capitalization). This new framework I called "**An Extended Knowledge Management Framework during Software Development Life Cycle".***

This paper highlights on knowledge management in software environments, its challenges, opportunities, implementation, and its success factors.

Keywords: *software development (SD), knowledge (K), knowledge management (KM), organizational memory (OM), requirements knowledge, domain knowledge, technical knowledge.*

1. INTRODUCTION

Software development is a collective, complex, and creative effort. In order to produce quality software, software organizations are trying to better use one of its most important resource: the organizational software engineering knowledge.

The demands on software development are increasing. Shorter time-to-market, better quality and better productivity are more and more goals to be achieved. To meet these requirements, software organizations have tried to better use one of its most important resource: the organizational software engineering knowledge.

Historically, this knowledge has been stored on paper or in people's mind. When a problem arises, we look for experts across our work, relying on people we know, or we look for documents. Unfortunately, paper has limited accessibility and it is difficult to update. In the other hand, in a large organization, it can be difficult to localize who knows some matter, and knowledge in people's mind is lost when individuals leave the company. Important discussions are lost because they are not adequately recorded. Therefore, knowledge has to be systematically collected, stored in a corporate memory, and shared across the organization. In other words, knowledge management is necessary.

In the context of software development, KM can be used to capture the knowledge and experience generated during the software process. Reusing knowledge can prevent the repetition of past failures and guide the solution of recurrent problems. Also, we cannot forget that collaboration is one of the most important knowledge sources for software organizations. But, to be effective in the software development context, a KM system should be integrated to the software process

Knowledge Management is an emerging discipline that promises to capitalize on organizations' intellectual capital. The concept of knowledge is far from new and phrases containing the word *knowledge* such as "knowledge bases" and "knowledge engineering" has been around for a while.

With an orientation to knowledge management in software development organizations, Davenport and Prusak describe knowledge as "a fluid mix of framed experience, values, contextual information, and expert insights and grounded intuitions that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of the knower. In software organizations, knowledge often becomes embedded not only in documents or repositories, but also in organizational routines, processes, practices, and norms" (Davenport and Prusak, 1998).

Software development is a complex set of tasks. It involves several scientific disciplines, like understanding the needs of other people, and technical issues like transferring requirements into a reliable and efficient computer program. It involves planning the process of developing the software, organizing work between several people, and sharing mental models on the status of the software in development. Software development is a discipline where one has to master both social and technical skills.

The first argument in favor of managing knowledge in software engineering is that it is a human and knowledge intensive activity (Birk, et. al., 1999).

But as software development projects grow larger and the discipline moves from craftsmanship to engineering, it becomes a group activity where individuals need to communicate and coordinate. Individual knowledge has to be shared and leveraged at a project and organization level, and this is exactly what KM proposes.

In software development one can identify two types of knowledge:
1. Knowledge embedded in the products (artifacts), since they are the result of highly intellectual, creative activities.
2. meta-knowledge that is knowledge about the products and processes

Software organizations are heavily dependent on tacit knowledge, which is very mobile" (Tiwana, 2000). If a person with critical knowledge about processes and practices suddenly leaves the organization, severe knowledge gaps are created (Brössler, 1999).
It is more important for Software Engineering organizations to exploit and manage their intangible assets in contrast to their physical assets" (Tiwana, 2000).

Software development organizations are knowledge-intensive firms where the knowledge is mainly embedded in human beings and is largely in the form of tacit knowledge.

This paper identifies the following critical knowledge areas:
- User requirements knowledge
- Functional domain knowledge
- Technical knowledge
- Project status knowledge
- Project experience knowledge.

2. RESEARCH PROBLEM

Software engineering is a knowledge intensive business and as such it could benefit from the ideas of knowledge management. The important question is, however, where does knowledge reside in software engineering?

It is clear that software engineering involves a multitude of knowledge-intensive tasks: analyzing user requirements for new software systems, identifying and applying best software development practices, collecting experience about project planning and risk management, and many others (Birk, et. al., 1999).

Software engineering is a complex business that involves many people working in different phases and activities.
The knowledge in software engineering is diverse and its proportions immense and growing. Organizations have problems keeping track of what this knowledge is, where it is, and who has it. A structured way of managing the knowledge and treating the knowledge and its owners as valuable assets could help organizations leverage the knowledge they possess.

3. KNOWLEDGE MANAGEMENT AND SOFTWARE ENGINEERING ENVIRONMENTS

Success in an increasingly competitive marketplace depends critically on the quality of the knowledge, which organizations apply to their business processes. The challenge of using knowledge to create competitive advantage becomes more crucial.

Software development is a collective, complex, and creative effort. As such, the quality of a software product heavily depends on the people, organization, and procedures used to create and deliver it. In other words, there is a direct correlation between the quality of the software process and the quality of the software developed [11].

4. THE NEED FOR CAPTURING AND SHARING PROCESS AND PRODUCT KNOWLEDGE

4.1. THE NEED FOR DOMAIN KNOWLEDGE

Software development not only requires knowledge about its own domain, but also about the domain for which software is being developed.
Domain knowledge that no one in the organization possesses must be acquired either by training or by hiring knowledgeable employees. Knowledge Management can, however, help organize the acquisition of new knowledge and it can help identify expertise as well as capture, package and share knowledge that already exists in the organization.

5. CHALLENGES FOR KNOWLEDGE MANAGEMENT IN SOFTWARE ENGINEERING

Implementing knowledge management in any organization is a challenge because of the time and effort that is required before it starts to return on the investment. Software organizations seem to have even less time than others because of the fast pace of the business.

Another challenge is the elusiveness of software. Unlike products of other domains, software is not visible (compare with buildings in the civil engineering domain) (Devanbu, et. al., 1990). Invisibility leads to less reuse of the system. Another result is that software developers are not accustomed to reuse, which is a problem because the idea behind knowledge management is reuse of assets.

The most problematic challenge to knowledge management is that most of the knowledge in software engineering is tacit and will never become explicit. It will remain tacit because there is no time to make it explicit. A way to address this problem can be to develop a knowledge sharing culture, as well as technology support for knowledge management, never forgetting that the main asset of the organization is its employees.

6. OPPORTUNITIES FOR KNOWLEDGE MANAGEMENT IN SOFTWARE ENGINEERING

It is clear that a knowledge management system needs to be supported by appropriate IT technology (Brössler, 1999). While IT technology can be intimidating to many people, this is not the case for software engineers (Schneider, 2001).

The other obvious benefit with software engineering activities is the fact that all artifacts are already in electronic form (Schneider, 2001) and, thus, can easily be distributed and shared.

7. KNOWLEDGE IN SOFTWARE ORGANIZATIONS

When individuals team up to solve a problem (or to develop a product), they form a ***community of practice***. When individuals communicate and exchange information related to a common topic, but for solving different problems within or outside a company, they form ***communities of interest***, such as groups of Java programmers. These communities heavily utilize web technology for knowledge sharing.

In software development, learning occurs during projects. For organizational learning, knowledge from all projects must be documented, collected and organized into a repository that will support decision making for future projects.

8. KNOWLEDGE MANAGEMENT IN SOFTWARE ENGINEERING

Knowledge management is seen as a strategy (or practice, systematic process, set of policies, procedures and technologies) that creates, acquires, transfers, brings to the surface, consolidates, distills, promotes creation, sharing, and enhances the use of knowledge (or information, intellectual assets, intellectual capital) in order to improve organizational performance; support organizational adaptation, survival and competence; gain competitive advantage and customer commitment; improve employees' comprehension; protect intellectual assets; enhance decisions, services, and products; and reflect new knowledge and insights.

It is clear that software engineering involves a multitude of knowledge-intensive tasks: analyzing user requirements for new software systems, identifying and applying best software development practices, collecting experience about project planning and risk management, and many others (Birk, et. al., 1999).

8.1. Knowledge Management support for core Software Engineering activities

The core software engineering processes and activities that might occur in a typical software engineering project are all documents (even the source code and the executable programs can be regarded as documents). (Birk, et. al., 1999). The work is, many times, focused on authoring, reviewing, editing, and using these documents.

Because software engineering is so dominated by the documents that are produced during the various activities and processes, the foundation for a knowledge management system is a ***document management*** system. Hand in hand with document management coming the need of distributing information about the project, which calls for general ***information management***.

8-2. Organizational Memory for Software Development

Learning from experience requires remembering history. Individual memory is, however, not sufficient and the entire organization needs a memory to explicitly record critical events. There are at least three distinguishable forms of organizational memory:

1. Memory consisting of regular work documents and other artifacts that were developed primarily to assist development of the product (examples in this category are requirements specification, and design specification)
2. Memory consisting of entities that were developed specifically to support the organizational memory (examples are lessons learned and post-mortem analyses)
3. A mix of the first two forms

9. Implementation of Knowledge Management

Implementing a KM system might, however, not be so simple, involving both challenges and obstacles. Examples of the most important issues noted by D. Rigby, analysts for Bain&Co. (Lawton, 2001) are:

- Technology issues:

KM involves software technology, but it is not always simple or even possible to integrate all the different subsystems and tools to achieve the level of sharing that was planned.

Inadequate security. While the idea behind KM is to share knowledge, it is important not to share knowledge assets with the wrong audience (e.g., competitors and former employees). This issue might limit the extent to which knowledge can be shared in the organization.

- Lack of standards:

Different parts of the organization might use terms and concepts in different ways. This lack of standards can inhibit sharing of knowledge between them.

- Organizational issues:

It is a mistake to focus only on technology and not on methodology. It is easy to fall into the technology trap and devote all resources to technology development without planning for a KM implementation approach.

- Individual issues

Employees do not have time to input knowledge or do not want to give away their knowledge.

10. An Extended Knowledge Management Framework during Software Development Life Cycle.

10.1. A Five C's knowledge management Life Cycle (By Al-khaldi, Alawneh, & khateeb, 2005)

- Knowledge ***c***apture phase
- Knowledge ***c***reation phase
- Knowledge ***c***odification phase
- Knowledge ***c***ommunication phase
- Knowledge ***c***apitalization phase

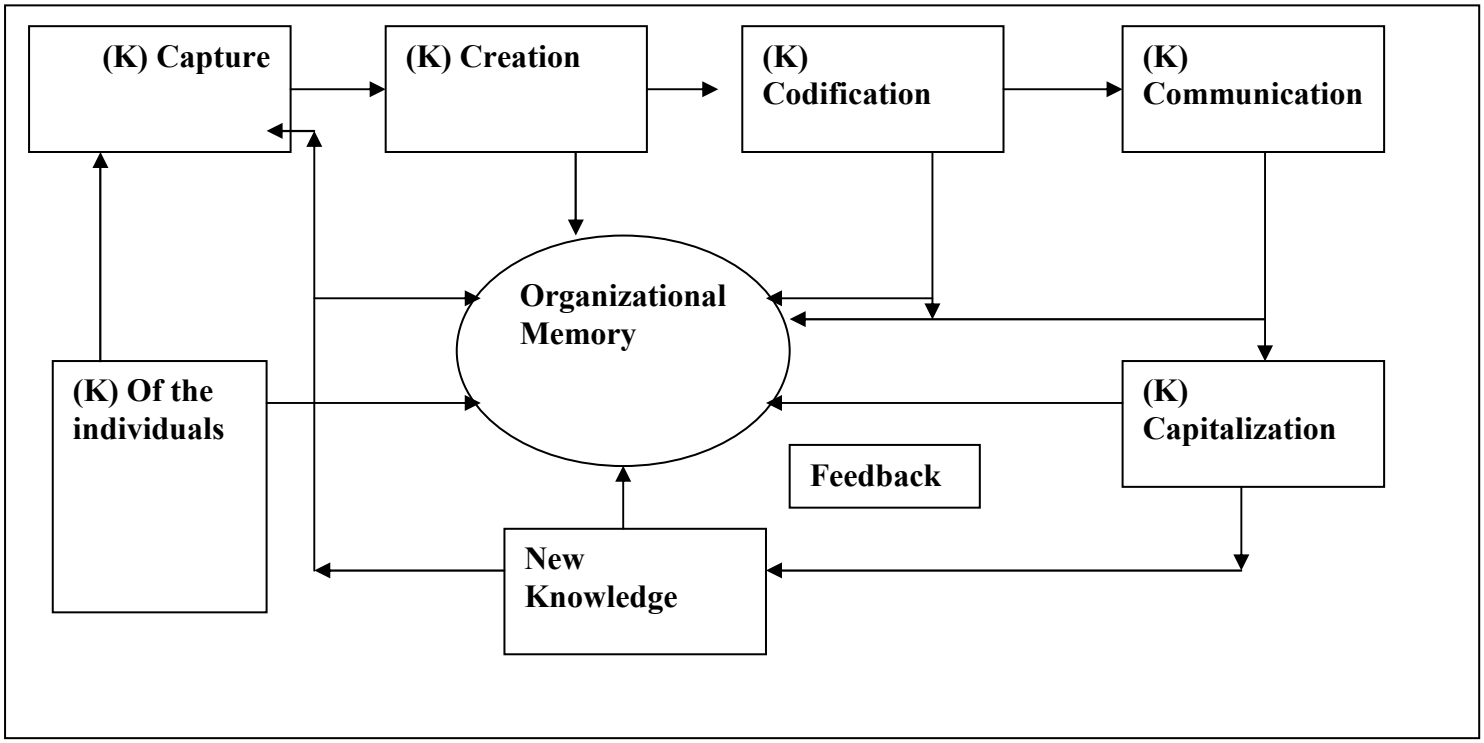


Figure1: A five C's knowledge management life cycle (Khaldi, Alawneh&khateeb, 2005)

10.2. A Five Layered knowledge management framework to model the software team knowledge

Software development is no longer a homogeneous field. A socio-technical approach and a commitment to project management principles are essential for attaining success in software development projects. But, managing project knowledge is another critical factor that has to be taken into consideration. Managing knowledge in globally distributed teams involves managing software projects' knowledge through the life-cycle of the development of the software project. The life-cycle of software development projects can be defined using the systems development life-cycle approach as shown in Figure2. Figure 2 also shows the various types of knowledge that needs to be managed during the project life-cycle. It has been observed that the following project related

Critical knowledge needs to be managed as the project progresses:

- User requirements knowledge
- Functional domain knowledge
- Technical knowledge
- Project status knowledge
- Project experience knowledge.

10.2.1 Need for Managing User Requirements Knowledge:

Meeting the client's requirements is critical to a software project's success; Clients may be unable to articulate their requirements. They may articulate the wrong requirements. Besides, different client groups may disagree over requirements. Their articulation of requirements may be misunderstood by the software developers. As a result of this and environmental volatility, requirements may change during a project. This uncertainty may lead to conflict, delays, cost over-runs, and failure to meet the client's needs. Requirements refer to the descriptions of properties, attributes, services, functions, and/or behaviors needed in the software to accomplish the goals and purposes of the system (Carr, 2000). At the system level, requirements should address the needs but should not specify a design solution. This should be left to the software designers in the team.

Thus, adopting a knowledge management perspective of requirements, Walz, Elam and Curtis (1993) make some specific recommendations for software project managers:

• Increase the amount of application domain knowledge across the entire software development team.

• Actively promote the acquisition, sharing, and integration of knowledge within a software design effort through team facilitation techniques and formally recognize these activities by allocating time to them.

- Much of the information that needs to become part of the team's memory is not captured formally, particularly, in standard documentation. Therefore, new tools (such as intranets) are needed to easily and unobtrusively capture this process-based information.

10.2.2 Need for Managing Technical and Functional Domain Knowledge

Knowledge from multiple technical and functional domains is a necessity for software development. This knowledge falls along at least three inter-dependent domains (Henninger, Lappala and Raghavendran, 1995):

• Application domain such as manufacturing, banking, transportation, etc.

• Technical domain

• Best practices in the two domains.

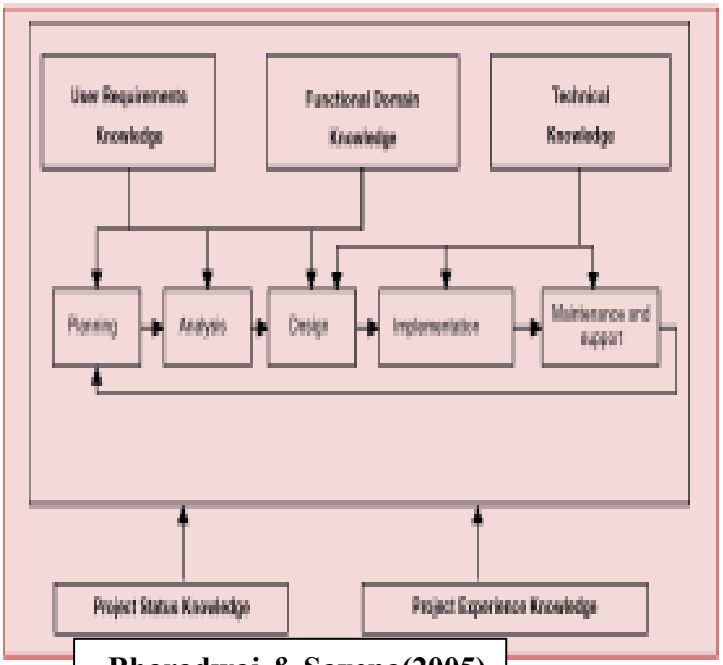
10.2.3 Need for Managing Project Status Knowledge

The third type of project knowledge, which must be available to the software team, is project status knowledge. Project documentation (such as requirements specification, design documentation, programme specifications, project plans, etc.) and standards (such as checklists, templates, standard procedures, etc.) need to be managed.

10.2.4 Need for Managing Project Experience Knowledge

Although issues are always project-specific, they may be having some generic patterns. Therefore, many of the issues encountered in a project could be relevant to other project sites or other projects as well. For example, the

Figure 2: Knowledge Areas during Systems Development Life-cycle

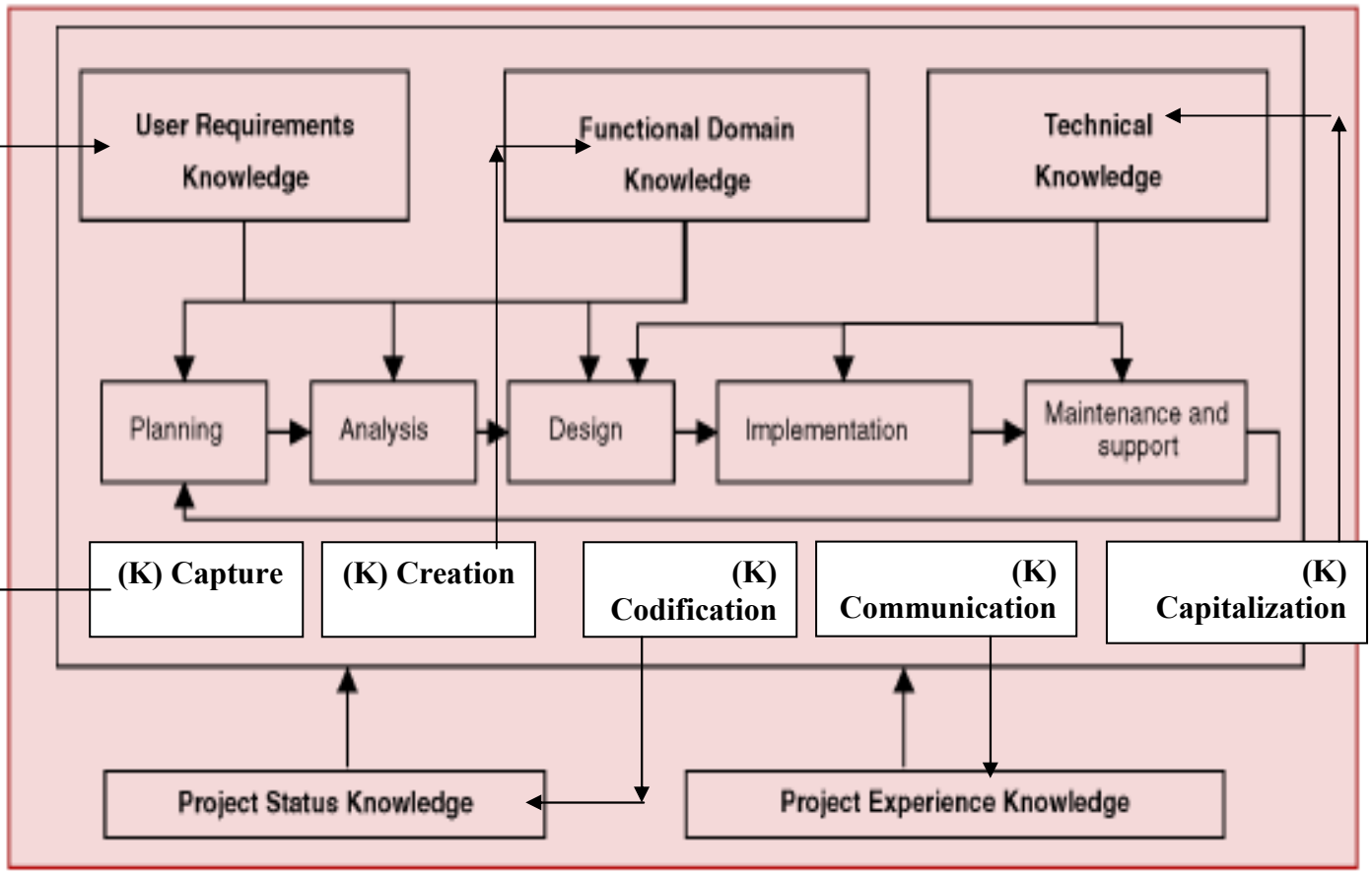


Bharadwaj & Saxena(2005)

Issues may be pertaining to important system requirements, instructions or clarifications for customers, innovative design ideas for addressing some problems, Precautions to be taken when using some software for development, etc.

10.3. An Extended Knowledge Management Framework during Software Development Life Cycle: A combination of Five Layered & Five C's Models

Figure3: An Extended Knowledge Management Framework during software development life cycle along with knowledge management life cycle. (**Alawneh, 2006**)



The above framework shows that in order to best utilize and harness the knowledge of employees during software development life cycle in software organizations. First, in planning phase, user requirements knowledge should be captured through Searching for several sources of knowledge that is necessary and related for performing the work , perceiving and sensing needs and requirements of work from knowledge resources , acquiring knowledge that already exist in organization from its appropriate sources at appropriate time where it is needed , extracting the knowledge of other people in organization , formulation of conceptual knowledge or idea (pre-mature vivid) from the knowledge that is available in organization , using metaphor mechanism in order to extract the hidden knowledge in organization , using brainstorming mechanism in order to solve the work problems of the organization .

then, functional domain knowledge should be created during analysis & design phases through Conducting research activities in order to discover the knowledge in the organization , exploitation from past experiences in the organization to discover new knowledge , creating new knowledge through the continuous learning in the organization , next, the project status knowledge should be codified throughout all phases of software development life cycle through Classification and categorization of existed knowledge in the organization according to its nature into categories such as administrative& technical & financial…etc, storing knowledge in organization in locations the are easily to retrieve , mapping knowledge in the organization so it can be easily to access whenever needed , organizing knowledge in the organization in a way so it is understandable to all organizational members, next, the project experience knowledge should be communicated among employees throughout all phases of development through Considering source , nature , and type of knowledge when transferring and sharing in the organization , motivating organizational members for participation in their creative and intellectual resources , encouraging and enhancing the culture of knowledge sharing among organizational members , providing information and communication technology in order to transfer knowledge among people in the organization, finally, the technical knowledge should be capitalized during implementation, maintenance and support phases through Investing and utilizing organizational knowledge in new ways and methods of doing work .

11. CONCLUSION

The focus of this paper is knowledge management in software engineering. It presents the developments in knowledge management in general, and for software engineering in particular, and discusses models for knowledge management. The paper also presents resources that can provide help, and information to software organizations that want to better manage their knowledge.

Software development is a knowledge- and people-intensive activity. Groups that are geographically distributed carry out a significant amount of the work in software engineering. People in such groups must collaborate, communicate, and coordinate their work, which makes knowledge management a necessity. However, for organizations that are large and distributed, whose environment is continuously changing, or have a high turnover, managing their knowledge assets is critical for survival.

Many businesses are human and knowledge intensive. Examples include consulting, advertisement, media, high-tech, pharmaceutical, law, software development, and other human capital-based organizations. Knowledge intensive organizations have realized that a large number of problems are attributed to un-captured and un-shared product and process knowledge, as well as the need to know 'who knows what' in the organization, the need for distance collaboration, and the need to capture lessons learned and best practices. These realizations have led to a growing call for knowledge management (KM).
A characteristic of software engineering that turns out to be an advantage over other industries in terms of managing intellectual capital is that artifacts are already captured in electronic form and can easily be stored and shared. In addition, software engineers often have a friendly attitude towards using new technology. This means that a software organization that implements a knowledge management system could have a good chance to succeed with this mission

In the context of software engineering, we define knowledge management as a set of activities, techniques, and tools supporting the creation and transfer of SE knowledge throughout the organization. One use of KM is to support software process improvement (SPI) activities. This support is important because both software engineering and quality management techniques fail if they are not based on a thorough knowledge of what is needed and what has been done in a software development organization.

Knowledge is the most powerful and ubiquitous resource of any organization. Any activity that does not leverage its power is clearly a sub-optimal utilization of the resources. Software development, a highly complex and intellectually intensive activity is not an exception. It involves intellectual effort by individuals in teams on projects with deadlines and deliverables that often change over the lifetime of the project. Fluctuating requirements and goals are occasioned both by greater clarity in clients' true requirements and constrains as the project progresses as also by promising new technologies that emerge and business exigencies that arise over time. The need to manage in such contexts is often why the software development process is characterized as undisciplined, chaotic and completely unpredictable.

This paper summarizes the status of the following critical knowledge areas:
 the most critical knowledge area is the user requirement knowledge. Though newer processes are introduced to manage the same, managing user requirements still remains a challenge for the members of the global software teams.
 Functional domain knowledge and technical knowledge are managed well by companies but technology updates have put pressure in identifying the gaps and bridging it during the project execution.
 Project status knowledge has been well managed in the global software teams with the help of formal procedures and documentation.
 Capturing and reusing the project experience knowledge of the existing projects and clients is still an open issue.

Some benefits of this extended framework can be pointed out:
• With KM integrated to software engineering environments, it is easier for developers to create new knowledge. In this way, the organizational memory is not closed. It is always evolving.
• A major concern for knowledge management in software development environment is to capture information during the software process without developers' extra effort. Thus, the KM system is actively integrated into the work process. An isolated KM system, on the other hand, can be a barrier to innovation, because it does not let workers share new ideas with their peers.
Closed systems do not give organizations control over their own knowledge, since there is a gap between knowledge creation and integration. Innovations happen outside the KM system, and then it contains information that is chronically out of date and that reflects an outsider's view of work.
• Knowledge management users are no longer passive receivers of knowledge, but are active researchers, constructors, and communicators of knowledge. Knowledge can be constructed collaboratively in the context of the work. Attention to knowledge requires attention to people, including their tasks, motivation, and interests in collaboration. The heart of intelligent human performance is not the individual human mind but groups of minds interacting with each other and with tools and artifacts.

REFERENCES

[1] Andrew B., "Using stakeholders, domain knowledge, and responsibilities to specify information systems' requirements". *Journal of organizational computing and electronic commerce*, 9(4), pp.287-296, 1999.

[2] Bharadwaj, s.s., & Saxena, B.k. "Knowledge management in global software teams". *Interfaces VIKALPA*, volume.30, no.4, pp.65-75, 2005.

[3] Birk A., Surmann D., & Althoff K., "Applications of knowledge Acqusition in Experimental Software Engineering", 11th European Workshop on knowledge Acquisition , modeling , and Management",pp.67-84, 1999.

[4] Brössler, P. "Knowledge Management at a Software Engineering Company - An Experience Report", *Workshop on Learning Software Organizations, LSO'99,* Kaiserslautern, Germany, pp. 163-170, 1999.

[5] Barbara P. "Project memories: integrating knowledge and requirements management". *Fraunhofer institute for experimental software engineering (IESE)*, Kaiserslautern, 2000.

[6] Conradi, R."From software experience databases to learning organizations". *International journal of software engineering and knowledge engineering*, vol.10, no.4, pp.541-547, 2000.

[7] Desouza, K.C. "Barriers to effective use of knowledge management systems in software engineering", *communications of the ACM*, vol.46, no.1, pp.99-101, 2003.

[8] Dai, et.al. "Software warehouse: its design, management and application". *International journal of software engineering and knowledge engineering*, vol.14, no.4, pp.395-406, 2004.

[9] Dingsoyr, T., & conradi, R." A survey of case studies of the use of knowledge management in software engineering". *International journal of software engineering and knowledge engineering*, vol.12, no.4, pp.391-414, 2002.

[10] D.E. O'Leary, "Enterprise Knowledge Management", *IEEE Computer Magazine*, March, 1998

[11] Davenport H., and prusak, L. *Working Knowledge*. Boston, Massachusetts: Harvard Business School Press, 1998.

[12] Hellstrom, T., Mikaelsson, J. "Decentralizing knowledge: managing knowledge work in a software engineering firm". *Journal of high technology management research*, 12(2001), pp.25-38, 2001.

[13] Jahnke, H., & Walenstein. "Evaluating theories for managing imperfect knowledge in human-centric database reengineering environments". *International journal of software engineering and knowledge engineering*, vol.12, no.1, pp.77-102, 2002.

[14] Komi-sirvio, S. et.al. "Toward a practical solution for capturing knowledge for software projects", *IEEE software*, may/June 2002, pp. 60-62, 2002.

[15] Lawton, G., "Knowledge Management: Ready for Prime Time" *IEEE Computer*, Vol. 34, No.2, pp.12-14, 2001.

[16] L.M.S. Borges, R.A. Falbo, "Managing Software Process Knowledge", *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'2002)*, pp. 227 – 232, Foz do Iguazu, Brazil, June 2002.

[17] Morasca, S., &Ruhe, G., "Knowledge discovery from empirical software engineering data". *International journal of software engineering and knowledge engineering*, vol.9, no.5, pp.495-498, 1999.

[18] Muller, C., Bahrs, J., Grohau, N.," Considering the knowledge factor in agile software development". *Journal of universal knowledge management*, vol.0, no.2, pp.128-147, 2005.

[19] Preece, A., et.al, "Better knowledge management through knowledge engineering, *IEEE Intelligent systems*, pp.36-43, 2001.

[20] Richter. H., & Abowd, G. "Tagging knowledge acquisition sessions to facilitate knowledge traceability", *International journal of software engineering and knowledge engineering*, vol.14, no.1, pp.3-19, 2004.

[21] Robillard, N. "The role of knowledge in software development". *Communications of the ACM*, vol.42, no.1, pp. 87-92, 1999.

[22] Rus, I. and Lindvall, M., "Knowledge Management in Software Engineering," *IEEE Software*, vol. 19, no. 3, pp. 26-38, 2002.

[23] Rus, Lindval, & Suman. "Knowledge management in software engineering". *Fraunhofer center for experimental software engineering (A state-of-the-art-report*). Rome, 2001.

[24] Schneider, K., "Experience Magnets - Attracting Experiences, Not Just Storing Them", *Product Focused Software Process Improvement, PrOFES'01*, Kaiserslautern, Germany, pp.126-140, 2001.

[25] Shaft, M., Michael, F., & Vessey, I." The role of cognitive fit in the relationship between software comprehension and modification". *MIS Quarterly*, vol.30, no.1, pp.29-55, 2006.

[26] Struder, R., etInvesting and utilizing organizational knowledge in new ways and methods of doing work , enhancing feeling of individual responsibility toward knowledge of organization , enhancing individual effectiveness though his acquiring of organizational knowledge.al. ,"Next generation knowledge management", *BT technology journal*, vol.23, no.3, 175-190, 2005.

[27] Tiwana,A., Mclean, E. ," Expertise integration and creativity in information systems development" .Journal of MIS, vol.22, no.1, pp.13-43,2005.