International Technology Management Review
Volume 1 Number 2 (November) 2008
<http://www.academicglobalpublications.com/itmr/>

An Extended Knowledge Management Framework during the Software Development Life Cycle

Ali A. Alawneh
Department of MIS
Faculty of Information Technology
Philadelphia University, Amman, Jordan
Email: alawneh2001@yahoo.com

Ezz Hattab
The Arab Academy for Banking and Financial Sciences (AABFS)
Faculty of Information Systems and Technology
Department of MIS, Amman – Jordan
Email: ehattab@aabfs.org

Walid Al-Ahmad
New York Institute of Technology, Amman
School of Engineering and Computing Sciences, Amman, Jordan
Email: Walid.AlAhmad@nyit.edu.jo

Abstract

Title: An Extended Knowledge Management Framework During the Software Development Life Cycle.
Keywords: *software development (SD), knowledge (K), knowledge management (KM), software engineering (SE), organisational memory (OM), requirements knowledge, domain knowledge, technical knowledge.*
Category of paper: Conceptual paper
Purpose of the paper: This paper describes the role of knowledge management and its application in the context of software development. Knowledge management (KM) can be used to capture, organize, and catalo knowledge and experience generated during the software process.
Methodology: Literature review
Findings: The paper proposes a new way of thinking about the role of (KM) in software engineering environments by developing an extended framework that integrates five types of knowledge into the five phases of software development lifecycle and the five phases of the KM life cycle. The results found that the proposed framework for managing knowledge during software development will help individuals in identifying the critical knowledge available during software development and choosing the right phase of KM lifecycle for the right knowledge area in the right phase of software development lifecycle.
Implications for practice: The application of the proposed framework will improve the success rate of software development projects, through enhancing the exchange and transfer of knowledge and experience among software development teams. This will also result in improved software development training programs, policies, project management, and practices in software development projects.
Value of the paper: The paper will be of interest to researchers, academics, knowledge workers, top management, and software practitioners.
Number of pages: 20
Number of figures: 4
Section headings: Abstract, Introduction, Research problem, Knowledge in Software Organisations, Reasons why people would not share knowledge, KM in Software Engineering Environments, The need for capturing and sharing process and product knowledge, A five C's KM lifecycle, A five-layered KM model for software team knowledge, Combining the five layered and the five C's models, Discussion and Key Findings, Conclusions and future work, References.

International Technology Management Review
Volume 1 Number 2 (November) 2008
<http://www.academicglobalpublications.com/itmr/>

An Extended Knowledge Management Framework during the Software Development Life Cycle

Software development is a collective, complex, and creative effort. Worldwide, there is an increasing demand for IT projects and the demand for skilled and experienced software developers is increasing as well. Shorter time-to-market, better quality and better productivity present the increasing number of goals to be achieved. To meet these requirements, software organisations have tried to better use one of their most important resources: the organisational software engineering knowledge.

Historically, this knowledge has been stored on paper or in people's minds. 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. On the other hand, in a large organisation, it can be difficult to locate who knows what, and knowledge in people's minds 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 the corporate memory, and shared across the organisation. However, knowledge is more than simply a list of things we know or a collection of facts. Therefore, knowledge management (KM) can play a vital role in encapsulating and spreading software development knowledge and expertise.

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 must not forget that collaboration is one of the most important knowledge sources for software organisations. But, to be effective in the software development context, a KM system should be integrated into the software development process.

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

With reference to KM in software development organisations, 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 organisations, knowledge often becomes embedded not only in documents or

repositories, but also in organisational 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 organisational level, and this is exactly what KM does.

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

Software organisations 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 organisation, severe knowledge gaps are created (Brössler, 1999). Therefore, it is more important for software engineering organisations to exploit and manage their intangible assets in contrast to their physical assets (Tiwana, 2000).

Software development organisations 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 five critical knowledge areas relevant to software development, namely user requirements knowledge, functional domain knowledge, technical knowledge, project status knowledge, and project experience knowledge.

Research problem

Software engineering (SE) is a knowledge-intensive business and, as such, it could benefit from the ideas of KM. The important question is, however, where and in what formats does knowledge reside in software engineering?

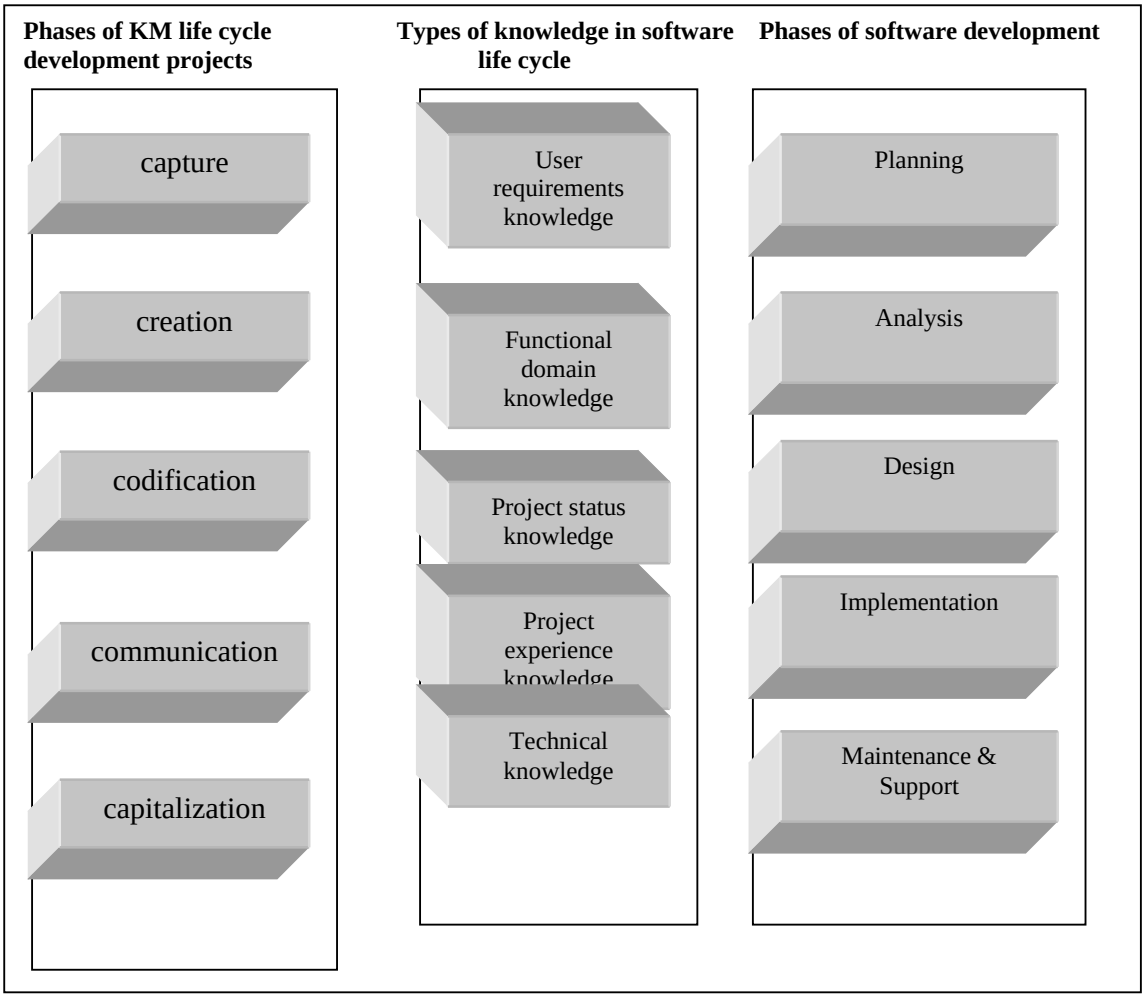
It is clear that SE involves a multitude of knowledge-intensive tasks: analyzing user requirements for new software systems; identifying and applying best software development practices; collecting experience about project planning and risk management; and many others (Birk, et al., 1999).

SE is a complex business that involves many people working in different phases and activities. The knowledge in SE is diverse and its proportions are immense and growing. Software organisations 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 organisations leverage the knowledge they possess. Therefore, there is a pressing need for a theoretically rigorous and empirically relevant framework for examining the use of KM in software organisations.

In sum, the research problem is defined as "developing an integrated model for knowledge management in software organisations including the phases of knowledge management lifecycle, types of knowledge required in software development projects, and phases of software development lifecycle (figure 1).

This model will help individuals who are working in software organisations to identify and catalog the different types of knowledge – that is available for software projects – to capture, create, codify, communicate and capitalize in different phases of software development.

Figure 1: The basic components of the proposed model



Source: Developed for this research

Knowledge in software organisations

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 organisational learning, knowledge from all projects must be documented, collected and organized into a repository that will support decision making for future projects (Schneider, 2001).

KM is seen as a strategy that creates, acquires, transfers, brings to the surface, consolidates, distills, promotes creation, sharing, and enhances the use of knowledge in order to: improve organisational performance; support organisational 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.
Implementing KM in any organisation is a challenge because of the time and effort that is required before there is a return on the investment. Software organisations 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 (compared with buildings in the civil engineering domain). 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 KM is reuse of assets.

The most problematic challenge to KM is that most of the knowledge in SE 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 KM, never forgetting that the main asset of the organisation is its employees.

It is clear that a KM system needs to be supported by appropriate IT infrastructure (Brössler, 1999). While IT 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.

Reasons Why People Would Not Share Knowledge

A company's culture reflects what people think and feel about the organisation. Do they trust each other and their management, and are they willing to go out of the traditional bounds of the work culture to benefit the organisation?
Software organisations need to realize that employees may feel possessive about their knowledge, and they may not be forthcoming in sharing it. After all, the knowledge they have is why they are valuable to the organisation, why they are paid by the organisation, and why they do not want to give that knowledge away. A term which is used these days is "capturing tacit knowledge", which is similar to "picking your employees' brains." This term sounds like software organisations are picking whatever their employees know. The "capturing emotion" might scare people into withholding their knowledge, thinking they will be expendable as soon as their employers have captured all of the knowledge they need. If this was the result of successful knowledge management, then everybody should be afraid of losing their job (Rus and Lindvall, 2002).

Here are several reasons why employees might be reluctant to share their knowledge: First, employees want the organisation to be dependent on them. If they share the knowledge with others, they fear they will loose their "expert" status. Second, some cultures encourage

individualism and ban cooperative work and sharing. In such cultures it is harder to establish a successful knowledge management program. As a matter of fact, most Western schools do not encourage students to work together in the classroom or while doing homework, so most students have learned that sharing is cheating. In order to create a sharing culture, such values and manners have to be unlearned. Third, employees might not be willing to share lessons learned because of their negative connotation. Lessons learned are based on incidents, some of which might be failures. Although the purpose is to learn from failures to avoid similar mistakes, many employees might fear that submitting negative lessons learned could be interpreted against them by management.

These are cultural issues that management must handle by creating a learning environment. Employees will, however, always be concerned with how management treats them, and the information that management has about them. Employees will react negatively if they fear that information will be used against them.

Knowledge Management in Software Engineering Environments

Success in an increasingly competitive marketplace depends critically on the quality of the knowledge, which organisations 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, organisation, and processes 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 (Davenport and Prusak, 1998).

1. First Level Knowledge Management: Knowledge Management Support for Core Software Engineering Activities

This section addresses core software engineering processes and activities. Birk illustrates the wide spectrum of software engineering processes that might occur in a typical software engineering project (Birk, et. al., 1999). What is common amongst the results from all these processes and activities is that they are all documents (even the source code and the executable programs can be regarded as documents). The work is, many times, focused on authoring, reviewing, editing, and using these documents. Due to the fact that many software organisations are distributed over large geographic areas, these documents need to be remotely available. 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.

Document management systems have been used for quite some time, but as the term knowledge management became popular, there was a tendency to re-label the document management tools as knowledge management tools, to accommodate the new trend. Portal technology enables web-based communication within or outside organisations. Although managing web sites can be fairly complicated (for example, they need support for links and content management) portals can certainly be valuable to software engineering projects that need to share knowledge captured in different forms.

As stated before, not all tacit knowledge in an organisation can be made explicit. Therefore, in order to fully utilize the competence of the organisation there is a need for keeping track of who knows what. Generally, employees do have knowledge about other employees' expertise

if the group is small enough (10-15 people), but larger groups of people are exposed to the risk of "not knowing what other people know." An elaborated solution to this problem is competence management (i.e., skills management or expert network).

Competence management systems were initially developed with the major objectives of being able to find employees with the right skills in order to staff new projects and to find individuals who have specific pieces of knowledge. Competence management has evolved over time into systems for much broader use.

2. Second Level Knowledge Management: Organisational Memory for Software Development

Learning from experience requires remembering history. Individual memory is, however, not sufficient and the entire organisation needs a memory to explicitly record critical events. There are at least three distinguishable forms of organisational memory: First, 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). Second, Memory consisting of entities that were developed specifically to support the organisational memory (examples are lessons learned and post-mortem analyses). Third, A mix of the first two forms.

3. Third Level Knowledge Management: Packaged Knowledge That Supports Knowledge Application

There are a large number of tools available, either as research prototypes or as commercial tools, that claim to be knowledge-based. Common for these tools is that they are specifically tailored for software engineering. They support the software engineer in his daily job and often result from analysis of knowledge from many previous projects.

The need for capturing and sharing process and product knowledge

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 organisation possesses must be acquired either by training or by hiring knowledgeable employees. KM can, however, enable the acquisition of new knowledge and it can help identify expertise as well as capture, package and share knowledge that already exists in the organisation.

The Need for Acquiring Knowledge About New Technologies
Knowledge Management fosters a knowledge sharing culture within the company that helps facilitate sharing of knowledge related to new technologies. Knowledge Management also makes the point that time should be spent on actively searching for knowledge both within the organisation and outside. Knowledge sharing occurs within communities of practice and interests, which can help speed up the learning curve.

The Need for Sharing Knowledge About Local Policies

Knowledge Management can help set up a system that encourages both informal knowledge sharing sessions and more formal ways of communicating. Lightweight knowledge management approaches attempt to capture the informal knowledge that is shared on a daily basis so that it can be disseminated on a larger scale.

The Need for Knowing Who Knows What

Much knowledge can be recorded, but, nevertheless, the assets of a software engineering organisation are mainly its employees and their tacit knowledge. Management of intangible assets includes knowing who knows what and is part of competence management. Knowing who knows what can help reduce the time it takes employees to find experts.

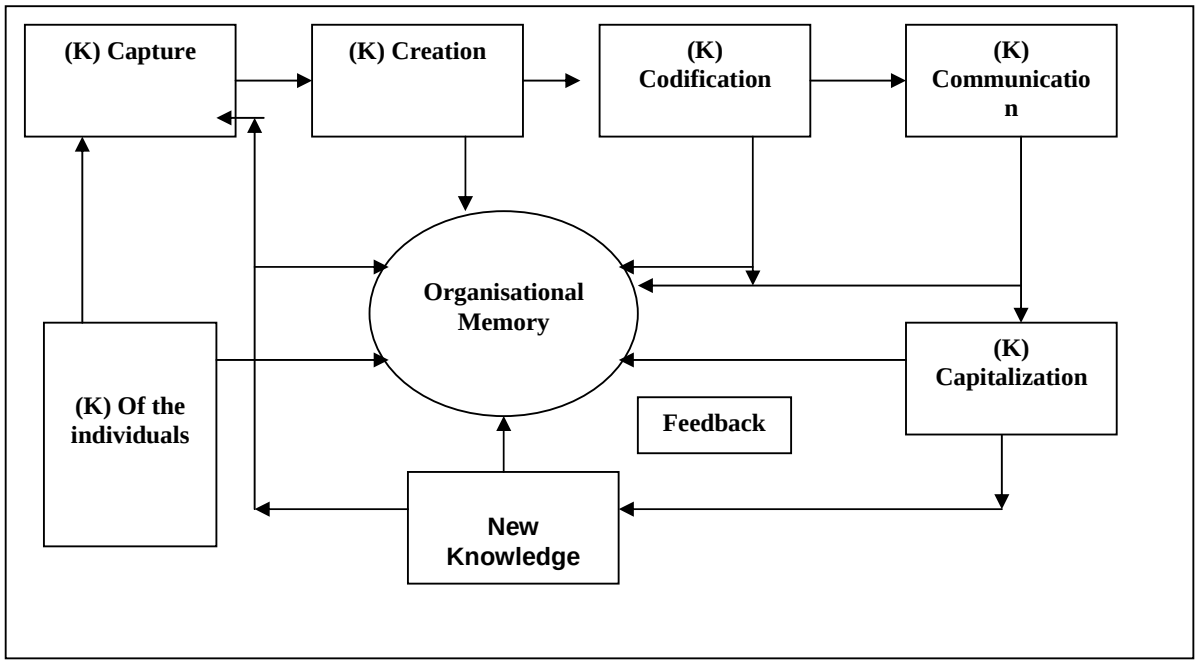
Knowledge Management can never tap the brains of the employees, but it can help build structures and frameworks for capturing key information that can help retain some knowledge when employees leave. This key information would at least help in understanding what the employee who left knew and what profile his successor needs to have to fill the position. Knowledge Management can help establish routines for identifying knowledge, as well as the people who own the knowledge--- the experts.

The Need for Distance Collaboration

Knowledge Management can help solve this problem as it acknowledges the need to capture, organize and store knowledge, as well as the necessity of knowledge transfer. Communication in software engineering is often related to the transfer of knowledge. Collaboration is related to mutual sharing of knowledge. Coordination that is independent of time and space is facilitated if the work artifacts and their status are stored and made part of an organisational memory.

A Five C's KM lifecycle

In (Al-khaldi, et al., 2005), the researchers proposed a new model of the KM lifecycle called the Five C's Model. The five Cs refer to: Capture-Creation-Codification-Communication-Capitalization. Figure 2 shows the five phases of the model, along with the tasks that are embedded in each phase. It also shows the interaction between the five knowledge phases.



(Source: Khaldi, et al., 2005)

Figure 2: The five C's knowledge management lifecycle

Next, we give a brief description of each knowledge phase:

Knowledge capture phase: the tasks that are embedded in this phase are: 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 exists in the organisation from its appropriate sources at appropriate times where it is needed, extracting the knowledge of other people in the organisation, formulation of conceptual knowledge or idea from the knowledge that is available in the organisation, using metaphor mechanism in order to extract the hidden knowledge in the organisation, using brainstorming to solve the work problems of the organisation, consulting others in the organisation to capture and acquire their knowledge, participation in training workshops and sessions in order to acquire more knowledge.

Knowledge creation phase: the tasks that are embedded in this phase are: conducting research activities in order to discover the knowledge in the organisation, exploiting past experiences in the organisation to discover new knowledge, creating new knowledge through the continuous learning in the organisation, preparing an appropriate culture and system in order to create new knowledge in the organisation, developing systematic knowledge in the organisation through combining explicit knowledge of people in the organisation, developing sympathetic knowledge in the organisation through socialization with other people in the organisation, developing new ways for doing work tasks in the organisation, referring to external consulting firms in order to discover new knowledge and new ways of doing, referring to departments and specialized units in the organisation to create knowledge, engaging and participating in work meetings with other people in the organisation in order to get common answers for work problems, enforcing strict conditions on people in the organisation to encourage them to create new knowledge and new ways of doing, forming

social networks of people in the organisation in order to let them generate new knowledge among one another, arousing states of uncertainty about knowledge during discovering of new knowledge, searching in the organisational setting for new knowledge through general routine procedures.

Knowledge codification phase: the tasks that are embedded in this phase are: classification and categorization of existing knowledge in the organisation according to its nature into categories such as administrative, technical, financial etc., storing knowledge in the organisation in locations that are easy to retrieve, mapping knowledge in the organisation so it can be easily accessed whenever needed, organizing knowledge in the organisation in a way that is understandable to all organisational members, considering the classification of the knowledge so it cannot be accessed except by authorized people, placing the knowledge of the organisation in suitable settings so it can be easily perceived and comprehended, knowledge of the organisation reflects what is actually known and done by the organisation, refining and filtering the knowledge of the organisation in order to access the most critical knowledge, providing the necessary mechanisms to simplify the expression and articulation of the knowledge from organisational members, distinguishing between explicit and hidden knowledge of people in the organisation, orientation of organisational members to available knowledge resources.

Knowledge communication phase: the tasks that are embedded in this phase are: considering source, nature, and type of knowledge when transferring and sharing in the organisation, motivating organisational members for participation in their creative and intellectual resources, encouraging and enhancing the culture of knowledge sharing among organisational members, providing information and communication technology in order to transfer knowledge among people in the organisation, taking into account that the power of the organisation is based on the extent of knowledge sharing among organisational members, encouraging dialogue, conversations, and discussions among people in the organisation in order to share in their knowledge, reaching common understanding of problems faced by the organisational members when performing their tasks, reaching collaborative group solutions through sharing their ideas, exchanging knowledge among people in the organisation through documents, manuals and catalogues, accessing knowledge in the organisation anywhere anytime when it is needed, determining who are the people who can transfer knowledge to them, determining mechanisms and methods for distributing and disseminating knowledge in the organisation.

Knowledge capitalization phase: the tasks that are embedded in this phase are: investing and utilizing organisational knowledge in new ways and methods of doing work, enhancing the feeling of individual responsibility towards the knowledge of the organisation, enhancing individual effectiveness through this acquisition of organisational knowledge, encouraging individual competitiveness through acquiring knowledge of the organisation, application of knowledge leading to changing organisational culture, application of knowledge leading to finding new managerial practices for performing organisational work, application of knowledge leading to making creative and intellectual resources available in the organisation, application of knowledge leading to improving overall performance of organisation, application of knowledge lead to balancing cost-benefit in the organisation through improving services and reducing costs, enhancing creative tasks and practices through the application of knowledge, improving the decision-making process and problem solving through application of knowledge, utilizing the knowledge that is embedded in procedures, rules, and norms in the organisation in order to direct the future behavior of organisational members, directing other people in the organisation for performing some roles and functions without transferring

knowledge to them, evolving organisational knowledge after its application through the feedback that results from evaluation tasks and functions of organisational members, evaluating the outcomes of organisational knowledge after its application through the services that are offered by the organisation.

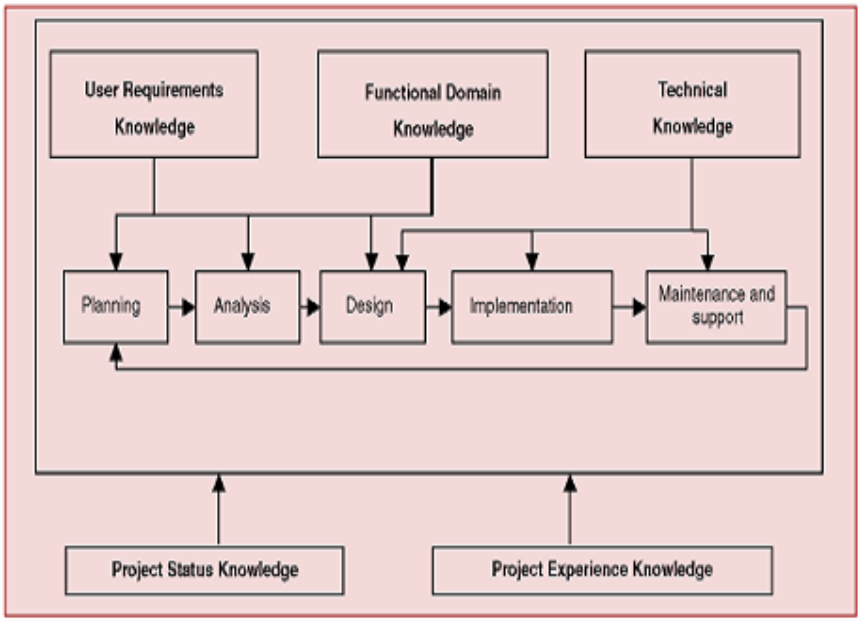
A five-layered KM model for software team knowledge

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. However, managing project knowledge is another critical factor that has to be taken into consideration. Managing knowledge in globally distributed teams involves managing software project knowledge through the lifecycle of the development of the software project. The lifecycle of software development projects can be defined using the systems development lifecycle approach as shown in Figure 3, which also shows the various types of knowledge that needs to be managed during the project lifecycle. It has been observed that the following five types of project-related critical knowledge need to be managed as the project progresses: User requirements knowledge, functional domain knowledge, technical knowledge, project status knowledge, and project experience knowledge.

Need for managing user requirements knowledge

Meeting the client's requirements is critical to a software project 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.

Figure 3: Knowledge Areas during Systems Development Life-cycle



(Source: Bharadwaj and Saxena, 2005)

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. 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 KM perspective of requirements is necessary. Some specific recommendations for software project managers are:

- 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.

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, namely the application domain such as manufacturing, banking, transportation, etc. the technical domain, and the best practices in the two domains.

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, program specifications, project plans, etc.) and standards (such as checklists, templates, standard procedures, etc.) need to be managed.

Need for managing project experience knowledge

Although issues are always project-specific, they may have 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 issues may pertain 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. Knowledge about the success and failure factors of projects is a valuable knowledge that should e managed properly to increase the success rate of software projects.

Combining the five layered and the five C's models

In order to best harness and utilize the knowledge of employees during the software development lifecycle in software organisations, the following steps are required as shown in Figure 4.

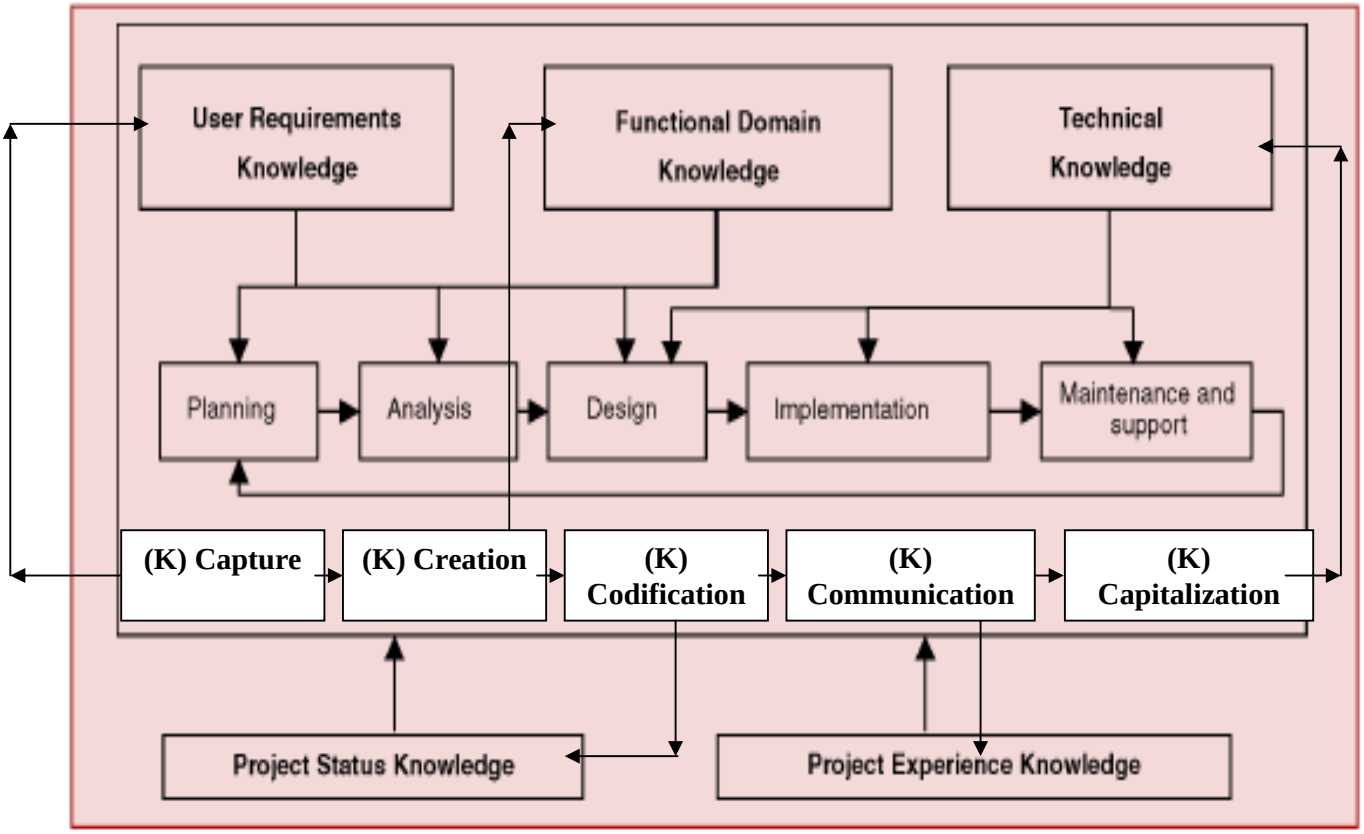
First, in the planning phase, the user requirements knowledge should be captured through searching for several sources of knowledge that is necessary and related for performing the work, perceiving and sensing needs and requirements of work from knowledge resources, acquiring knowledge that already exists in the organisation from its appropriate sources at the appropriate time where it is needed, extracting the knowledge of other people in the organisation, formulation of conceptual knowledge or idea from the knowledge that is available in the organisation, using a metaphor mechanism in order to extract the hidden

knowledge in the organisation, using a brainstorming mechanism in order to solve the work problems of the organisation.

Second, functional domain knowledge should be created during analysis and design phases through conducting research activities in order to discover the knowledge in the organisation, exploitation of past experiences in the organisation to discover new knowledge, creating new knowledge through the continuous learning in the organisation.

Third, the project status knowledge should be codified throughout all phases of the software development lifecycle through classification and categorization of existing knowledge in the organisation into categories such as administrative, technical, financial, etc., storing knowledge in the organisation in locations the are easy to retrieve, mapping knowledge in the organisation so it easy to access whenever needed, organizing knowledge in the organisation in a way that is understandable to all organisational members.

Fourth, the project experience knowledge should be communicated among employees throughout all phases of the development through considering the source, nature, and type of knowledge when transferring and sharing knowledge in the organisation, motivating organisational members to share their creative and intellectual resources, encouraging and enhancing the culture of knowledge sharing among organisational members, providing information and communication technology in order to transfer knowledge among people in the organisation.



(Source: Developed for this research)

From the above proposed model, three main categories for software engineering tasks are identified:

1. Tasks performed by a team focusing on developing a software product based on customer requirements.
2. This represents the core task of any software organisation. The team leader (project manager) is responsible for ensuring that work is completed on time and within budget and possesses the intended functionality and quality. Software engineering is document-oriented and what is produced during the project is a set of documents such as contracts, project plans, requirements and design specifications, source code, test plans and related documents. These documents are not just work products. There is also additional information embedded within them: (1) during the project they document the decisions; (2) after the project's completion, they contain the history of the project. The documents can be reused in different ways by the next project so that people can learn from them, by analyzing the solutions to different problems that these documents capture.
3. Tasks that focus on improving a team's ability to develop a software product.
   Here we can include tasks that might be conducted during and shortly after the project. The reason for performing these tasks is to ensure that potential knowledge gained in the project is not lost. Included here are all forms of lessons learned and post-mortem analyses that identify what went right or wrong in the project. Also included are analyses of data from the project, for example, comparisons of budgeted and actual costs, estimated and actual effort, planned and actual calendar time. Tasks in this category attempt to collect and create knowledge about one particular project. The results from this activity are useful by themselves, but can also be the basis for further learning. They can be stored in repositories and experience bases (for example, in lessons learned repositories).

4. Tasks that focus on improving an organisation's or an industry's ability to develop software.

   This category represents activities that analyze results from several previous projects in order to identify similarities and differences between them. The insights gathered by these analyses can be formulated as knowledge or experience packages and can be qualitative, quantitative, or a mix of both.

Discussion and Key Findings

Knowledge intensive organisations have realized that a large number of problems are attributed to un-captured and unshared product and process knowledge, as well as the need to know 'who knows what' in the organisation, the need for remote collaboration, and the need to capture lessons learned and best practices. These realizations have led to a growing call for KM.

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

A characteristic of SE 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 positive attitude towards using new technology. This means that a software organisation that implements a KM system could have a good chance to succeed with this mission.

In the context of SE, we define KM as a set of activities, techniques, and tools supporting the creation and transfer of SE knowledge throughout the organisation. One use of KM is to support software process improvement (SPI) activities. This support is important because both SE 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 organisation.

Knowledge is a valuable resource of any organisation. 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 the 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.

Some benefits of the extended framework include:

- With KM integrated into software engineering environments (SEEs) where a knowledge management system needs to be supported by appropriate IT technology to produce artifacts in electronic form, that can easily be distributed and shared. With this integration between KM and SEEs, it is easier for software developers to create new knowledge. In this way, the organisational memory is not closed. It is always evolving.
- A major concern for KM in the software development environment is to capture information during the software process without extra effort on the part of the developers. 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 organisations 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.
- KM 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.

As pointed earlier, the most critical knowledge area is the user requirement knowledge. Though newer processes are introduced to manage requirements, managing user requirements still remains a challenge for the members of the global software teams. Although functional

domain knowledge and technical knowledge are managed well by companies, yet technology updates have put pressure in identifying the gaps and bridging them during the project execution. Project status knowledge has been well managed in the global software teams with the help of formal procedures, checklists, and documentation. However, these are incomplete and inadequate in most cases. Also, capturing and reusing the project experience knowledge of the existing projects and clients is still an open issue. Therefore, these areas can indeed benefit from the new approach to integrating knowledge management into the full software development lifecycle.

The extended framework can provide software engineering organisations with a set of factors for a successful implementation of a KM system. First, it provides a knowledge friendly culture where software organisations values learning and innovation, and establishes appropriate incentives and reward systems. Employees collaborate and have a positive attitude towards knowledge. When there is free flow of knowledge from other employees, individuals tend to respond in the same manner. Second, it places employees in an environment where they have opportunities to use their capabilities to the fullest. Third, it motivates employees to share their knowledge with other people in the organisation. They must be convinced that their sharing of knowledge will be valuable to the organisation and, most importantly, to themselves. Fourth, it develops a broadly shared understanding of the organisation's mission, current direction, and the role of the individual in support of the organisation and of the individual's own interests. Fifth, it manages the knowledge base the same way as physical assets. Time and effort should be invested in designing, building and maintaining its content. Sixth, it links all knowledge management systems to other information systems. Seventh, it provides the continuous monitoring, evaluation, and guidance of the KM activities and their plans, results and opportunities. Eighth, it create problem-solving groups comprised of people from a variety of disciplines. This will transfer the knowledge from one discipline to another, as well as provide solutions to interdisciplinary problems in decreased time. Nineth, it provides multiple channels for knowledge transfer, as each adds value in a different way. It is particularly important to provide opportunities for face-to-face contact, as well as electronic forms of communication. Tenth, it gives employees permission to innovate, improvise and stretch organisation policies and practices beyond the predetermined scopes.

Conclusions and future work

The focus of this paper is on how SE can benefit from developments in KM. Knowledge is the most powerful and ubiquitous resource of any organisation in general and software development organisations in particular. Therefore, integrating the lifecycles of both SE and KM is useful and necessary. This paper introduced a new approach to such integration. The proposed framework is based on the five C's KM model, which defines and relates five major KM processes, and the five-layered model, which defines and relates the main five software development activities that can best benefit from KM. We believe that the new framework can have a positive impact on the software industry in terms of the success rate of IT projects.

The efforts to identify and catalog knowledge constructs for software development activities are still in their infancy. It is hoped that this work will invite a wider population of software developers and knowledge managers to help refine and validate the model introduced in this paper.

As future work, we will be investigating the practical aspects and the effectiveness of the proposed framework by carrying out a pilot project. Currently, we are investigating several issues related to this pilot project such as the application domain (e-commerce or traditional), organisation (government or private), project size (small or medium), software development process (waterfall or iterative and incremental), etc.

References

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

Barbara P. 2000, 'Project memories: integrating knowledge and requirements management', Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern.

Bharadwaj S., Saxena K. 2005, 'Knowledge management in global software teams', Interfaces VIKALPA, volume.30, no.4, pp.65-75.

Birk A., Surmann D., Althoff K. 1999, 'Applications of knowledge Acqusition in Experimental Software Engineering', 11th European Workshop on Knowledge Acquisition, Modeling, and Management, pp.67-84.

Borges S., Falbo A. 2002, '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.

Brössler P. 1999, 'Knowledge Management at a Software Engineering Company – An Experience Report', Workshop on Learning Software Organisations, LSO'99, Kaiserslautern, Germany, pp. 163-170.

Conradi R. 2000, 'From software experience databases to learning organisations', International journal of software engineering and knowledge engineering, vol.10, no.4, pp.541-547.

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

Davenport H., Prusak L. 1998, Working Knowledge, Boston, Massachusetts: Harvard Business School Press.

Davies J., et al. 2005, 'Next generation knowledge management', BT technology journal, vol.23, no.3, 175-190.

Desouza C. 2003, 'Barriers to effective use of knowledge management systems in software engineering', communications of the ACM, vol.46, no.1, pp.99-101.

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, vol.12, no.4, pp.391-414.

Hellstrom T., Mikaelsson, J. 2001, 'Decentralizing knowledge: managing knowledge work in a software engineering firm', Journal of high technology management research, 12(2001), pp.25-38.

Jahnke H., Walenstein A. 2002, '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.

Khaldi F., Alawneh A., Khateeb A. 2005, 'A five C's knowledge management lifecycle', Faculty of Information Systems and Technology, AABFS, Working Paper.

Komi-Sirvio S., et.al. 2002, 'Toward a practical solution for capturing knowledge for software projects', IEEE software, vol. 19, no. 3 pp. 60-62.

Lawton G. 2001, 'Knowledge Management: Ready for Prime Time' IEEE Computer, vol. 34, no.2, pp.12-14.

Morasca S., Ruhe G. 1999, 'Knowledge discovery from empirical software engineering data', International journal of software engineering and knowledge engineering, vol.9, no.5, pp.495-498.

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

O'Leary E. 1998, 'Enterprise Knowledge Management', IEEE Computer Magazine.

Preece A., et.al. 2001, 'Better knowledge management through knowledge engineering', IEEE Intelligent systems, pp.36-43.

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

Robillard N. 1999, 'The role of knowledge in software development', Communications of the ACM, vol.42, no.1, pp. 87-92.

Rus I., Lindvall M. 2002, 'Knowledge Management in Software Engineering', IEEE Software, vol. 19, no. 3, pp. 26-38.

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

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

Tiwana A., Mclean E. 2000, 'Expertise integration and creativity in information systems development', Journal of MIS, vol. 22, no. 1, pp. 13-43.