Return On Experience About Implementing Knowledge Management Systems in Software Engineering: Motivations, Opportunities and Challenges

Mohammed Amine Mostefai [1], Mohamed Ahmed-Nacer [2]

[1]Ecole Supérieure d'Informatique (ESI), Algiers, Algeria
m_mostefai@esi.dz

[2]USTHB, Algiers, Algeria
anacer@mail.cerist.dz

Abstract—*The implementation of Knowledge Management Systems (KMS) in a long and hard process that requires resources, time and budget.*

As a return of an experience of a project of an implementation of KMS in a software engineering organization, this paper synthetizes the motivations, the opportunities and the challenges that have to be considered in such a project.

This paper focuses also on the solutions that we have adopted during our experience to address the challenges of this kind of projects.

Index Terms—*Software Engineering, Knowledge Management Knowledge Management System, Motivations, Barriers, Challenges, Opportunities*

1. Introduction

Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [1]. SE involves many collaborators working in different phases and activities.

During these phases, the collaborators produce and consume knowledge. This knowledge is various, growing and hardly traceable: organizations have problems keeping track of what this knowledge is, where it is, and who owns it [2]. Knowledge Management is a promising approach that addresses these problems.

Knowledge management is a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company's knowledge [3].

To achieve an optimal usage of knowledge, KM relies on four main processes: knowledge discovery, knowledge capture, knowledge sharing and knowledge application [4].

To support the KM processes, KM relies on a technological infrastructure called Knowledge Management Systems (KMS) [5].

KMS utilize a variety of technologies and mechanisms to manage, track and implement these processes.

A good implementation of KM in software engineering requires putting a robust KMS infrastructure in place, which is not an easy task.

KMS can be implemented in any organization of any domain (such as software engineering) that needs an optimal access and usage of knowledge.

However, each domain has its own specificities that can hamper or encourage the KMS implementation.

In this perspective, our team have led an experience of implementing a KMS in a software engineering organization [6].

As a synthesis of our work and our state of the art, this paper discusses the particularities of software engineering that could be challenges to take up or opportunities to seize when a software development organization plans to implement a KMS.

Thus, this paper is structured as follows: section 2 is about the motivations of implementing KMS in SE. Sections 3 exposes some key concepts related to KM and KMS, the main feature of KMS and some work in literature related to implementing KM in software engineering.

Sections 4 points some opportunities related to the SE domain that can encourage the KMS implementation while section 5 concerns the challenges that should be taken in consideration when implementing KMS in SE and the solution that we adopted to address them. Finally, section 6 concludes.

2. Motivations

All software engineering activities are activities that make an extensive usage of knowledge. The direct consequence of that fact is that knowledge grows significantly as long as the projects and activities are running.

Thus, the primary motivation of implementing KMS in the organization is to provide the necessary infrastructure that improves the usage and the access to that knowledge.

We can summarize the motivations of implementing KMS in a SE environment in the following points [2]:

2.1. Decrease time and increase quality

Increase the product quality and reduce the delays and the costs is the core purpose of software engineering discipline itself. On the hand, KM is a good way to reach these objectives: if KM is optimally applied in an organization, we can avoid error repetition and promote success factors reproduction. Moreover, knowledge captured during past projects can be very beneficial to the upcoming ones.

2.2. Acquire knowledge about new technologies

As any technological environment, the SE environment is an environment where technology is evolving at a hallucinating rythm.

It is very difficult to maintain a balance between respecting the project delays and learning about new technologies related to that project.

KM is a good channel that allows the acquisition of knowledge concerning new technologies quickly and efficiently.

2.3. Access domain knowledge

Developers do not only spend time in learning about new technologies, the acquisition of the domain knowledge can also be a time-consuming activity.

KMS provides an infrastructure that facilitates the transfer of the acquired domain knowledge between developers. It allows for example, new hired developers to share the understanding of the domain of their ancient colleagues in an acceptable time.

2.4. Quickly adopt the organization culture

Most of SE companies have a set of well-defined processes, policies, practices and culture.

KM provides a mean that allow new developers to adopt in a reasonable delays the cultural requirements of their new company and integrate efficiently the active development teams.

2.5. Skill identification

Big and middle-sized software development organizations are composed of hundreds to thousands of employees. Often, these organizations do not know that a particular person masters a particular technology or has a particular skill.

If implemented well, KMS will provide the right tool that permits to identify the right person to the right task in an organization.

3. BACKGROUND

Before discussing about the challenges and the opportunities, it is vital to understand some key concepts related to KM and KMS, introduce the main feature of KMS and quote some work related to the implementation of KM in software engineering. This is the purpose of this section.

3.1. Key concepts

Knowledge is defined as a justified belief that increases an entity's capacity for effective action [7].

There are two types of knowledge: (1) the explicit knowledge which is a formal or semi-formal knowledge taking the form of documents, manuals, formulas, etc., and (2) tacit knowledge which is a deeply individual knowledge. While the first type of knowledge is easy to share and transmit, the second type is difficult to formalize and transmit.

Knowledge is subject to four possible transformations [7]: (1) the externalization concerns the formalization mechanisms that allow transforming individual knowledge into sharable knowledge, (2) the internalization process consists of learning by transforming a formal knowledge to a personal knowledge, (3) the combination process is about composing multiple formal knowledge sources to create a new formal knowledge or experience and (4) the socialization process consists in combining informal knowledge without passing by any formalization.

Knowledge management is a method that simplifies the process of sharing, distributing, creating, capturing and understanding of a company's knowledge [3].

Knowledge Management Systems are a class of information systems designed for sharing and integrating knowledge [8]. KMS are the technological infrastructure that supports the KM in the organization.

3.2. The KMS Main Features

The main features of a KMS are categorized in six main categories [9]: (1) the content repository feature allows the storage and the retrieval of knowledge, (2) the collaboration feature is about the communication tools permitting to link users, (3) the knowledge dissemination feature concerns the mechanisms of a quick and an efficient access to knowledge, (4) the content integration feature concerns importing knowledge from external sources, (5) the domain ontology feature targets the representation of the domain knowledge and (6) the knowledge security feature is how the KMS protects knowledge and secures the access to it.

3.3. Related work

The community of software engineering has shown a growing interest in implementing KM in the last decade [10].

One of the most famous achievements in implementing KM in software engineering (SE) is the experience factory brought by Basili et al. in [11].

In [12], authors detail some case studies of implementing KM in some software companies. When studying the implementation of KM in SE, the work of Rus and Lindvall [2] is an important foundation. In [2], the authors highlight the motivations and the managerial aspects of implementing KM in SE.

In [6], we have proposed a methodology for the development of KMS targeting Component-Based Software Engineering.

Although, there is an obvious interest in implementing KM in software engineering, more focus is given to the managerial and strategic aspect of KM than the infrastructural one.

Our objective, through this paper, is to discuss the challenges and the opportunities of implementing KM in software engineering from an infrastructural point of view.

4. OPPORTUNITIES

This section lists some SE specificities that could be a very encouraging factors to a successful KMS implementation.

4.1. An important part of SE knowledge is explicit

The biggest difficulty when implementing KM in organizations is to handle the tacit knowledge that is stuffed in the brain of individuals.

The infrastructure aspect of KM, incarnated in the KMS, is oriented to handle the explicit knowledge while the tacit knowledge is handled by the managerial and strategic aspects of KM.

Fortunately, in software engineering a considerable amount of the knowledge assets are already explicit. These assets can be documents, plans, reports or even source code which can be a very important representation of an explicit knowledge asset [13].

4.2. The software engineering environment is a KMS-friendly environment

To implement KMS, many technical requirements are needed such as servers, hardware equipment and an acceptable bandwidth.

Most of middle-sized and large-sized development companies do have the adequate environment to host the KMS.

Besides, in some domains, the users are not accommodated with an extensive usage of computers and this can hamper the implementation of the KMS in their environment. Fortunately, this is not the case of SE where collaborators make an extensive usage of computers.

4.3. Many CASE tools are extensible

One of the most important issues that have to be considered when implementing KMS is its integration with existing tools.

In SE, many of the CASE (Computer-Aided Software Engineering) tools have an extensible architecture that allows the integration of third-party plugins. Eclipse [14] and Visual Studio [15] are examples of that extensible architecture.

This open architectural choice permits the integration of new plugins that make the tools that the development team is accustomed with, integrates easily with the KMS.

4.4. KMS are not necessarily developed from scratch

Often, the KMS are not built from scratch but are developed by the extension of an existing system [6]. For example, this system can be a collaborative system or a document management system.

One of most attractive opportunities is that many software development organizations have already such a system in their environment. The efforts of implementation can be considerably reduced as these systems already offer many required KMS features such as storage, versioning or search engines.

Moreover, the adaptation of these systems requires the development of some add-ons to adapt them to be a KMS. While developing these add-ons could be real barriers to other companies, this is not the case for SE organizations where development is the core activity.

4.5. Research is converging to a SE ontology

The construction of an ontology related to SE is an important objective that many organizations and laboratories target.

Some interesting works that address developing ontologies for SE can be found in the literature [16, 17]. The other encouraging factor is that many of the emerging ontologies are built based on the SWEBOK [18] that has a large agreement in the SE community.

4.6. The knowledge sharing culture is well-established in the software engineering community

Without its users, the KMS is useless. The more these users share their knowledge using the KMS, the more the other users access and apply it, and more KM is effective in the organization.

The sharing culture is incarnated implicitly and explicitly within software organizations. Thus, organizations do emphasize on some practices such as source code repositories, document and file repositories and also some meetings such as lessons-learned meetings. These practices are an explicit manifestation of the sharing culture.

The sharing culture goes even outside the organization boundaries.

For example, some websites such as The Code Project [19] or Stack Overflow [20] are real gold mines to most of developers.

On the other hand, the open source model constitutes another implicit channel of knowledge sharing. Developers have access to millions of code in sites such as Source Forge or Google Code [21].

Moreover, the sharing culture was one of the principal motivations of the emerging social software engineering (SSE) discipline [22]. One of the most interesting examples of SSE in the real world is the website geeklist [23] which contains at the moment of writing this paper, more than 20000 subscribers.

5. CHALLENGES

Implementing a KMS is a project that is expensive and complex.

In addition to some general challenges [8, 24] that can be categorized in technological challenges (infrastructure, hardware,…) and managerial challenges (people involvement, culture), SE has some specificities that can become a barriers to the KMS implementation.

We list in this section the most relevant challenges specific to the implementation of KMS in a SE organization and how we proceeded to address them during our experience.

5.1. Software Engineering is a vast domain

One of the main requirements of a KMS is a conceptualization of knowledge that allows its storage and utilization. Ontologies and taxonomies are a natural way to achieve this [9, 25].

Nevertheless, software engineering is a vast domain structured around many activities such as requirements analysis, design, implementation and tests where each activity involves many collaborators having various profiles [26].

Building a conceptualization that targets such a domain is a real obstacle as it needs estimable efforts of development and validation.

However, this challenge should not be an obstacle to start the KMS project. One of the solutions is the progressiveness in the KMS implementation.

For example, in our experience [6], instead of targeting the whole domain, we covered some aspects about a sub-domain of SE which is component-based software engineering (CBSE).

Another encouraging point, is that as stated in §4.5, research works are converging to a consensual SE ontology that makes the conceptualization already done.

5.2. Convince software engineers to use the KMS

Having a KMS put in place is not an objective by itself. The real objective is not only to implement it, but it is, above all, to make it usable and useful.

However, the project managers' nightmare is not the knowledge creation and sharing but it is the driving of the project according to the expected delays.

Consequently, the development team is usually an overloaded team that is more focusing on the project than on the knowledge about the project.

In such conditions, it is very difficult to convince collaborators to consecrate some time to share knowledge and to use the KMS.

One of the solutions is to exploit existing habits of developers instead of overriding them. In [6], we built the KMS by extending the existing intranet so the KMS was not something really new for developers and its utilization was not a time-consuming task.

5.3. The KMS impacts are difficult to measure

Because the implementation of KMS require time, resources and consequently, money, it is vital that some metrics have to be developed to measure the impact of the KMS in increasing the product quality and / or decreasing the delays and costs.

However, developing metrics that aim at measuring these effects are hard [8] and consequently, the return on investment evaluation is difficult to estimate.

Linking the impacts of the KMS in the quality, costs and delays related to the software products is a complex and a long-running process.

One of the solutions to address this challenge is to put in place task-based measures. For example, in [6], one of the main measure that we adopted is the quality of KMS-based component selection. Another measure, is the project-based measure that compares the metrics of the conductions of projects (costs and delays) using the KMS to the earlier metrics before the implementation of KMS.

5.4. The domain knowledge is difficult to integrate

Software development can target infinite and various domains such as finance, banking, education, healthcare or gaming.

In addition to the difficulty due to the complexity of the SE domain, the implementation of the KMS should address the issue of representing the domain knowledge knowing that it could be completely heterogeneous from a project to another.

In [6] we have implemented the KMS by the extension of an existing system called the base system. One of the features of this system is that it allows easily the modelling of entities using columns and content types. We used this same mechanisms to model both CBSE-oriented concepts and domain concepts.

5.5. Technology-related knowledge obsolescence

As mentioned in section 2.3, the SE faces a special kind of technology-related knowledge that is subject to a very quick obsolescence.

Knowledge valid for a certain version of a product or programming language could be very quickly obsolete when later versions are released.

The KMS should provide the mechanisms that ensure that the accessed knowledge is still up-to-date to make it useful to the current development project.

One of the solution that we adopted in [6] is meta-data. For any piece of knowledge that was shared in KMS, a certain number of metadata should be given such as the used technology and the tools versions. This choice helps us to evaluate knowledge according to the current tools and technologies.

5.6. The KMS integration

In software engineering, the KMS will be implemented in an environment that contain a plethora of other tools such as configuration management tools, editors, compilers, bug trackers, etc. These tools are called CASE tools (Computer-Aided Software Engineering) tools [26].

The integration of the KMS with these tools is not an easy task and should be taken in consideration when the organization plans to implement the KMS.

Hopefully, as we stated in §4.3, many case tools are open and some of them provide APIs to ease the integration process. In our experience in [6], we have integrated a source repository with the KMS using the API of that product.

5.7. The KMS should be secure

The security consideration is not something particular to SE but concerns every domain characterized by sensitive knowledge.

However, many development companies distinguish themselves in a very competitive market by their know-how concerning a particular product or service. If this knowledge is intentionally or unintentionally accessed by competitors due to a security lack in the KMS, the consequences would be fatal to the company.

One of the solutions is to use the security foundation of the KMS if this KMS is developed on the top of an existing base system. For example, in [6], the KMS security foundation was built on Active Directory security system. We used this prebuilt system to secure our KMS.

5.8. The KMS should support software processes

In software engineering, a software process is a set of activities whose goal is the development or evolution of software [26].

These processes are organized and executed according to a higher level representation called software process models (SPM). SE includes a large variety of SPMs such as the waterfall model or the spiral model [27].

The KMS should be capable to support these process model including their specificities and requirements.

On the other hand, many SE organizations are adopting agile methodologies to conduct their development projects. One of the elements of the agile manifesto is “Working software over comprehensive documentation” [28].

This manifesto element seems to be contradictory to the objective of the KMS to capture the developer knowledge in the most comprehensive manner possible. The KMS implementation project should take this element in consideration if the organization is using and applying agile methodologies.

In [6], the KMS targets an organization that use agile methodologies for their developments. The KMS was integrated well with the software processes as it was hosting all the documents related to the iterations such as the product backlogs and the iteration backlogs. All the meetings minutes were shared using the KMS. The test plans and results were also shared and versioned in the KMS.

6. Conclusion

Using knowledge effectively in software development organizations is vital to ensure their competitiveness and survival. KM is a discipline that targets the optimal capture, access, transfer and application of knowledge. KMS are a special kind of information systems that support the KM processes.

In our paper, we discussed the opportunities and the challenges that have to be considered when implementing KMS in a software engineering organization.

In addition to the general challenges, SE is characterized with some specificities that could make the KMS implementation even harder. These specificities are related to the conceptualization of the SE domain, the motivation of the development team to use an additional tool, the difficulty to measure the KMS impact in software products, the conceptualization of domain knowledge and the support to software processes.

In a previous experience of a project of the implementation of a KMS in a software engineering organization, we have presented how we addressed these challenges using the extension of an existing collaborative system.

Besides, our work has shown that SE has some other specificities that are real opportunities to implement KMS.

The explicit nature of an important part of SE knowledge assets, the SE environment that is KMS-friendly, the extensibility of CASE tools, the development of KMS by the extension of existing systems, research in SE ontologies and the sharing culture in the developers community are one of the most relevant of these opportunities.

References

[1] IEEE, "IEEE Standards Collection: Software Engineering," ed: IEEE Standard 610.12—1990, 1993.

[2] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE Software,* vol. 19, pp. 26-38, 2002.

[3] T. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*: {Project Management Institute}, 1997.

[4] I. Becerra-Fernandez and R. Sabherwal, *Knowledge management: systems and processes*: M.E. Sharpe, 2010.

[5] M. Alavi and D. E. Leidner, "Review : Knowledge Management and Knowledge Management Systems : Conceptual Foundation and Research Issues," *MIS Quarterly,* vol. 25, pp. 107-136, 2001.

[6] M. A. Mostefai and M. Ahmed-Nacer, "An Agile Methodology For Implementing Knowledge Management Systems : A Case Study In Component-Based Software Engineering," *International Journal of Software Engineering and Its Applications,* vol. 5, 2011.

[7] I. Nonaka and N. Konno, "The concept of "Ba": building a foundation for knowledge creation," *California Management Review,* vol. 40, pp. 40-54, 1998.

[8] M. Alavi, R. H. Smith, and D. E. Leidner, "Knowledge management systems: issues, challenges, and benefits," *Communications of the AIS,* vol. 1, 1999.

[9] W. Staniszkis, "Feature Requirements of a Knowledge Management System," Rodan Systems S.A.2003.

[10] F. Bjornson and T. Dingsoyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," *Information and Software Technology,* vol. 50, pp. 1055-1068, 2008.

[11] V. R. Basili, G. Caldiera, and H. D. Rombach, "THE EXPERIENCE FACTORY," *Encyclopedia of Software Engineering,* vol. 2, 1994.

[12] T. Dingsøyr and R. Conradi, "A survey of case studies of the use of knowledge management in software engineering," *International Journal of Software Engineering and Knowledge Engineering,* vol. 12, pp. 391–414, 2002.

[13] P. Klint and C. Verhoef, "Enabling the creation of knowledge about software assets," *Data Knowl. Eng.,* vol. 41, pp. 141-158, 2002.

[14] Eclipse. (2012, 2012). *The Plug-in Development Environment (PDE)*. Available: http://www.eclipse.org/pde/

[15] Microsoft. (2012, 2012). *Extend Visual Studio*. Available: http://msdn.microsoft.com/en-us/vstudio/ff718165

[16] O. Mendes and A. Abran, "SOFTWARE ENGINEERING ONTOLOGY: A DEVELOPMENT METHODOLOGY," *Metrics News,* vol. 9, pp. 68-76, 2004.
[17] C. Calero, F. Ruiz, and M. Piattini, *Ontologies for software engineering and software technology*: Springer, 2006.
[18] A. Abran, P. Bourque, R. Dupuis, J. Moore, and L. Tripp, *Guide to the Software Engineering Body of Knowledge - SWEBOK*: IEEE Press, 2004.
[19] (2012). *The Code Project*. Available: http://www.codeproject.com/
[20] (2012). *Stack Overflow*. Available: http://stackoverflow.com/
[21] *Google Code*. Available: http://code.google.com/
[22] N. Ahmadi, M. Jazayeri, F. Lelli, and S. Nesic, "A Survey of Social Software Engineering," presented at the SE Workshops '08: 23rd IEEE/ACM Intl. Conf. on Automated Software Engineering, Washington, DC, 2008.
[23] (2012). *Geek list*. Available: http://geekli.st
[24] C. S. Choy, "Critical Factors In The Successful Implementation Of Knowledge Management," *Journal of Knowledge Management Practice,* 2005.
[25] R. Dieng-Kuntz, *Corporate Semantic Webs*, 2011.
[26] I. Sommerville, *Software Engineering*: Addison-Wesley, 2006.
[27] R. S. Pressman, *Software engineering: a practitioner's approach*: McGraw-Hill Higher Education, 2010.
[28] S. Augustine, *Managing Agile Projects*: Prentice Hall; illustrated edition, 2005.