Capturing Software-Engineering Tacit Knowledge

ABDULMAJID HISSEN MOHAMED
Department of Computer Science
Sebha University
PO Box 18758, Sebha
LIBYA

Abstract: - Knowledge management systems are aimed to provide knowledge-intensive organisations with tools, and methods to better manage their knowledge capital. A great success is gained in the course of managing explicit knowledge in the form documented knowledge fragments. But, greater part of organisations knowledge is realised in tacit form which is volatile and hardly captured in a formal way. Managing this type of knowledge still represents one of the major challenges in knowledge management research. This paper proposes a knowledge model that caters for capturing both tacit and explicit organisational knowledge in the software-engineering domain.

Key-Words: - Knowledge Management, software, tacit knowledge, knowledge reuse, software artefacts

1 Introduction

As a result of the emerging knowledge-based economy (or K-Economy), knowledge has become the strategic resource of organizations to compete and survive. In line with this trend, the introduction of knowledge management systems (KMS) is receiving greater attention. KMS aims at providing an organization the ability to benefit from its past experience in order to respond more effectively in present situations. Organizational knowledge could be distinguished as formal (explicit) and informal (tacit) knowledge. According to Conklin [4], *Formal knowledge is the stuff of books, manuals, documents, memos, white papers, plans and training courses, whereas informal knowledge is the knowledge that is created and used in the process of creating the formal results. It includes ideas, facts, assumptions, questions, guesses, stories, points of view, etc*. Notice that these forms of tacit knowledge are personal and only reside in experts' heads. In short, it constitutes what Koskinen [8] describes as the practical know-how.

Traditional KMS has mostly focused on the collection and dissemination of explicit (*formal)* knowledge, which is structured and stored in electronic knowledge repositories. But, there is a growing recognition that tacit knowledge constitutes an essential part of the organisational memory, and it has to be considered as well. According to Zack [15], explicating tacit knowledge so it can be efficiently and meaningfully shared and reapplied, especially outside the originating community, is one of the least understood aspects of knowledge management. Few attempts have been made to manage or at least partially capitalise tacit knowledge, but these efforts suffer limitations in managing it adequately. Capturing tacit knowledge is more complex than the management of explicit knowledge for multiple reasons. For example, tacit knowledge is highly personal and organisations have no control on its sharing and exploitation. Also there are many psychological as well as technical barriers that work against capturing the full richness of tacit knowledge.

The rest of the paper is organised as follows: first, the importance of capturing tacit knowledge is highlighted. Second, an overview of technical and non technical challenges of capturing tacit knowledge is discussed. Third, the conceptual framework and components of the proposed model is explained. The paper ends with a conclusion and suggestion for further work.

2. The importance of tacit knowledge

Experience is an indispensable resource for any organization to perform better in handling upcoming similar situations. Unfortunately organizations do not control all knowledge assets they have. Great portion of organizational knowledge is realized in tacit form which usually resides in workers' heads as personal belief (know-how). Unfortunately when a human expert becomes unavailable either temporarily or permanently, then his/her expertise becomes unavailable too. In many situations knowledge workers who have been ousted for

various reasons, are called back for consultation. This situation simply occurs because part of the organisation's knowledge is taken away and it has to be retained. In a worst scenario, the knowledge expert might move to one of the business competitors, which means partially disarming knowledge of the former employer. This is why in organizations where workers turnover is high such as software engineering, the codification of tacit knowledge becomes an extremely important issue. The importance of managing tacit knowledge is well expressed in the following quote by *Cummings* [3], "*An organization's knowledge walks out of the door every night – and it might never come back".* Unfortunately, recent huge investment in IT is focused more on the management of explicit knowledge with less emphasis on the part of tacit knowledge management. Johannessen et al [6] asserts that companies easily lose their competitive edge if they emphasis investment in, and use of IT without taking tacit knowledge into consideration.

3 Barriers to capturing tacit knowledge

3.1 Non-Technical obstacles

Knowledge management is not a technology problem [14, 10]. Since it involves the externalisation of human expertise, human factors highly influence the success of any KMS. These non-technical factors are mostly related to firstly, how to characterise human knowledge, and secondly, how to promote knowledge workers to proactively contribute to externalise their tacit knowledge in order to build a group memory system. Usually there is a tendency that knowledge workers are very reluctant to share their personal knowledge with others. Some workers regard their professional knowledge as a source of job security and they are often reluctant to share it. Other organisational practices might also contribute to workers reluctance to share knowledge. According to Kokkoniemi [7], in organisations where staff can be rewarded for having particular expertise, this might lead to staff do not want to share what they know. In addition, psychological problems might also influence the position of knowledge workers participation in KM settings. For example, in meetings where the tacit knowledge usually produced and reproduced, some workers hesitate to share their knowledge and this would adversely influence the effectiveness of meetings as a result. Becker [1] justifies this attitude by, "reluctance of some members to speak, due to their shyness, low status, laziness or controversial ideas" [1]. All these problems have its roots in human psychological behaviour. Therefore, we believe that the externalisation of tacit knowledge has to be carried out as part of a defined knowledge lifecycle. It should not be left as merely voluntarily act. For example, in software engineering, deliberations held in different reviews of software projects lifecycle including the post-mortem reviews have to be formally captured. Unfortunately, the importance of the integration between software process models and knowledge lifecycle has probably not yet been sufficiently understood.

3.2 Technical Obstacles

In addition to non technical obstacles that contribute to difficulties in recognising and capturing tacit knowledge, tacit knowledge is usually less structured and it does not conform to a standard format. Therefore, it can be hardly coded and captured using traditional data models. From the software engineering viewpoint, design rationale models such as DRL [9], QOC [11] and IBIS [13] are all aimed to codify tacit knowledge in the form of different alternatives and justifications behind developers' design decisions. In spite of the support of capturing tacit knowledge as facilitated by these models, they cannot accommodate the full richness of tacit knowledge. Sometimes, people are expected to articulate their tacit knowledge differently; they might even resort to the use of body language. There is no one model that can accommodate all these variations in expressing tacit knowledge. Another major limitation shared by all of these models is that, they are not coupled well with explicit knowledge in the form of knowledge embedded artefacts. We believe knowledge artefacts should be regarded as incubators of tacit knowledge.

The widespread use of the Internet made email systems an easy way to capture tacit knowledge. This approach can be exemplified by TeamInfo[2]. But, this approach still has limitations in the form of the lack of face to face interactions where richer tacit knowledge is likely to surface. Another major problem of email-based approaches lies in the fact that, the captured knowledge (in the form of email messages) is not sufficiently accessible by computational means because they do not conform to a standard knowledge model (i.e. free text). Therefore, we believe there is a need for knowledge models that provide decent characterisation of tacit knowledge and to integrate both tacit and explicit knowledge fragments. The target models also have to cater for managing the

uptodatedness of captured knowledge based on a defined lifecycle. These requirements we believe will have potential to make our model workable.

4 The proposed knowledge model

A model is a purposeful abstraction of some part of reality [5]. It is meant to encapsulate the conceptual framework of the reality being modeled. Basically, our model is built around the notion of Knowledge Asset (K-Asset) as the basic building block of the potential software-engineering knowledge repository. K-Assets represent the smallest level of granularity of the knowledge repository. From the perspective of the software engineering domain, any useful fragment of software development knowledge can be regarded as a K-Asset. For example, any *lesson learned* or *process model* or *knowledge-embedded software artefact* can be considered as K-Asset. Individual knowledge workers can propose what they consider as valuable software engineering K-Assets. These K-Assets are then scrutinized by peer knowledge workers subsequently.

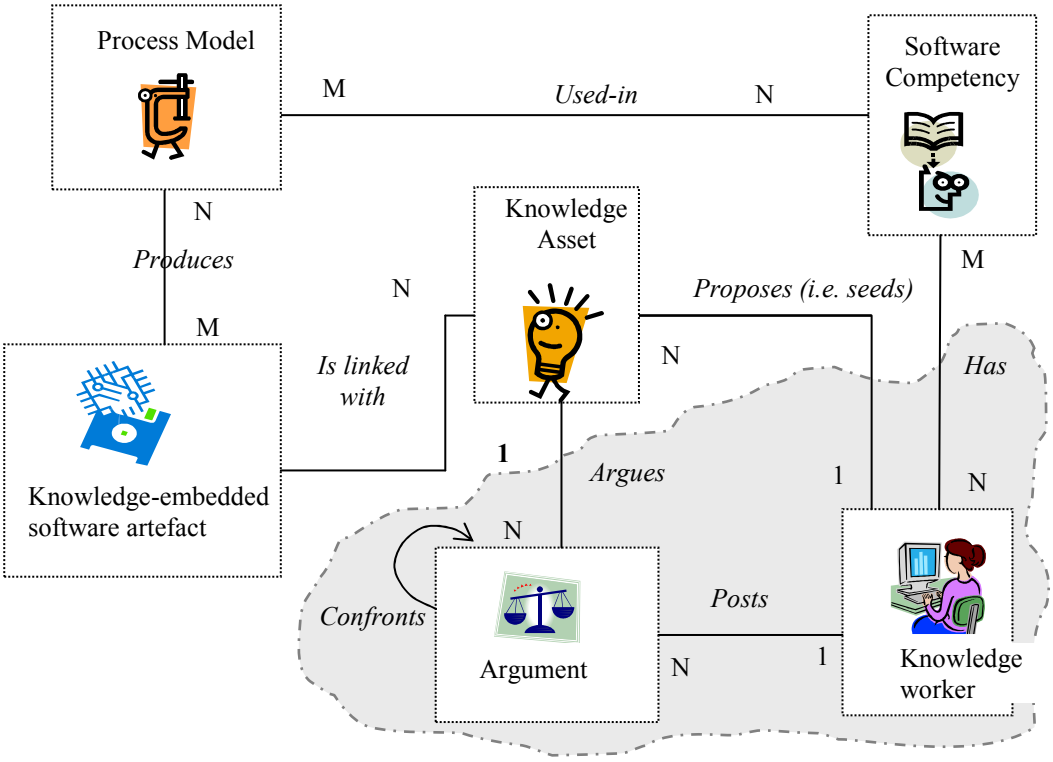

Fig. 1 Conceptual view of the software-engineering Knowledge Model

Figure 1 shows the higher-level abstraction of the proposed knowledge model. It shows the stages of the lifecycle of software engineering knowledge. Based on the proposed knowledge model, the knowledge generation loop starts from characterizing the building blocks of the experience knowledge. We believe the building blocks of the software experience are the *knowledge workers* and the set of *software/hardware competencies* they master and the *knowledge-embedded software artefacts* they produce. Any *knowledge-embedded software artefact* is in fact generated using one or a combination of software development skills in the form of software/hardware competencies. As a middleware between the *knowledge workers* and the *knowledge-embedded software artefacts* is the *process models* that needs to be followed. For example, in the case of any C programmer (*i.e. knowledge worker*), S/he must have mastery level of C programming (*i.e. software competency*). S/he then can follow the structured programming concept (*i.e. process model*) in order to produce any piece of software code. Supposing that the produced software code is an error trapping routine since this reusable code has the potential to be embedded in any C based software. This code can then be characterized as reusable knowledge fragment (*i.e. knowledge-embedded software artifact*).

Of course nothing can be characterized as a "knowledge fragment" simply based on the assessment of individual knowledge workers. Sometimes, the characterization of what can be considered as knowledge asset is a controversial issue because different knowledge workers might disagree about the validity of certain knowledge assets. What might be proposed by a knowledge worker as a knowledge fragments might not be qualified as such by others who might be more knowledgeable. Moreover, the changing of circumstances and newly emerged facts might influence the context of previously characterized knowledge assets. This is the reality of the knowledge phenomena anyway. For example, at one point of time, Ptolemy's theory (c. 2nd century CE) that we see objects because of the rays of light emanating from the eyes was very acceptable. With time and greater understanding, Ptolemy's theory was contradicted by ibn al-Haytham (c. 965-1040), who is regarded to be the father of modern optics. ibn al-Haytham correctly argued that humans see objects because the sun's rays of light are reflected from the objects into our eyes. This proves the fact that knowledge validity is dynamic. Therefore, based on the proposed model, the process of qualifying the proposed knowledge fragments relies on the consensus of other knowledge experts. This consensus is realized through other experts' opinions *(i.e. arguments*) in relation to qualifying or disqualifying the proposed K-Assets. These arguments includ qualifying/disqualifying criteria posted by peer knowledge workers make up the context associated with the captured knowledge. This collaborative knowledge filtering constitutes a pool for generating tacit knowledge in the form of views, suggestions, and all types of arguments (see the shaded portion in figure 1).

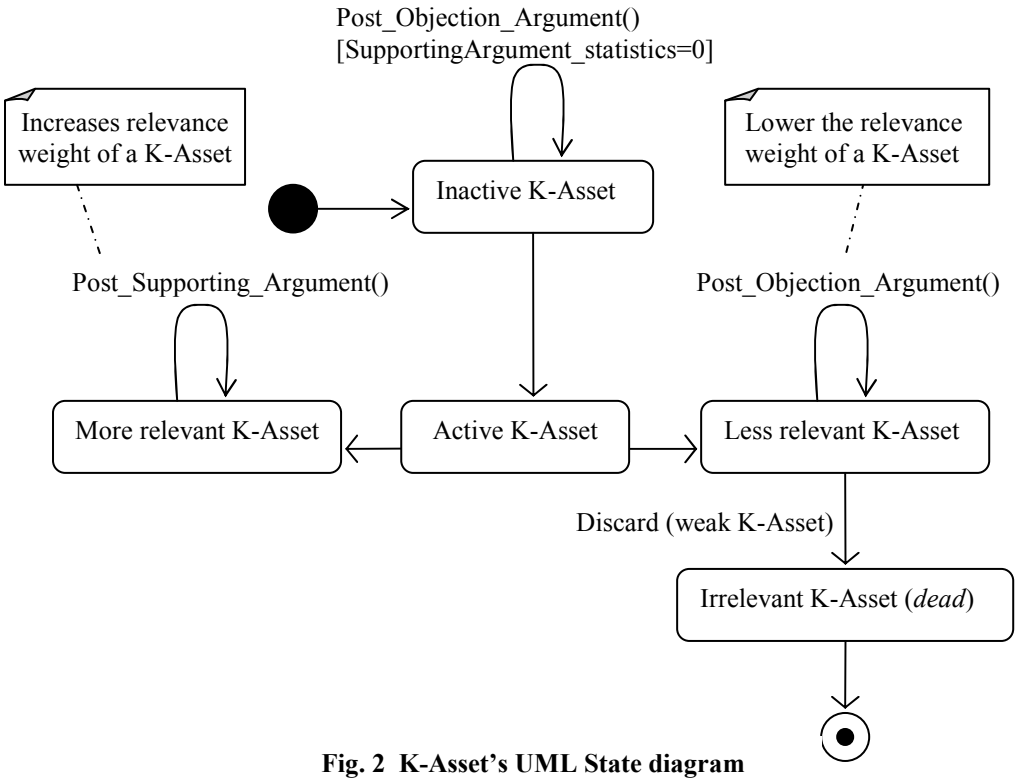

Fig. 2 K-Asset's UML State diagram

Since, knowledge may become useless when the situation changes. Therefore, any captured knowledge has to be continuously examined and the relevance status of individual K-Assets changes accordingly. This goes inline with Nonaka's claim that a company is not a machine but a living organisation [12]. We also believe that an organisation is living in the sense that it has to continuously examine what it knows and change its business beliefs accordingly.

In the case of the proposed knowledge model, tacit knowledge generated through the collaborative knowledge filtering helps to maintain the up-datedness of the captured knowledge. Figure 2 shows UML state chart diagram that demonstrates how the captured tacit maintains the dynamism of the relevance of captured K-Assets. Newly proposed

K-Assets are considered inactive and only become relevant as it receives the agreement and support from other knowledge workers. Likewise, the more objections it receives the less relevant it becomes. The model discussed has been implemented through LiSER which is a software-engineering knowledge management system. It is meant to capture not only explicit knowledge but also tacit knowledge in the form of frequent arguments posted by knowledge workers (see Figure 3).

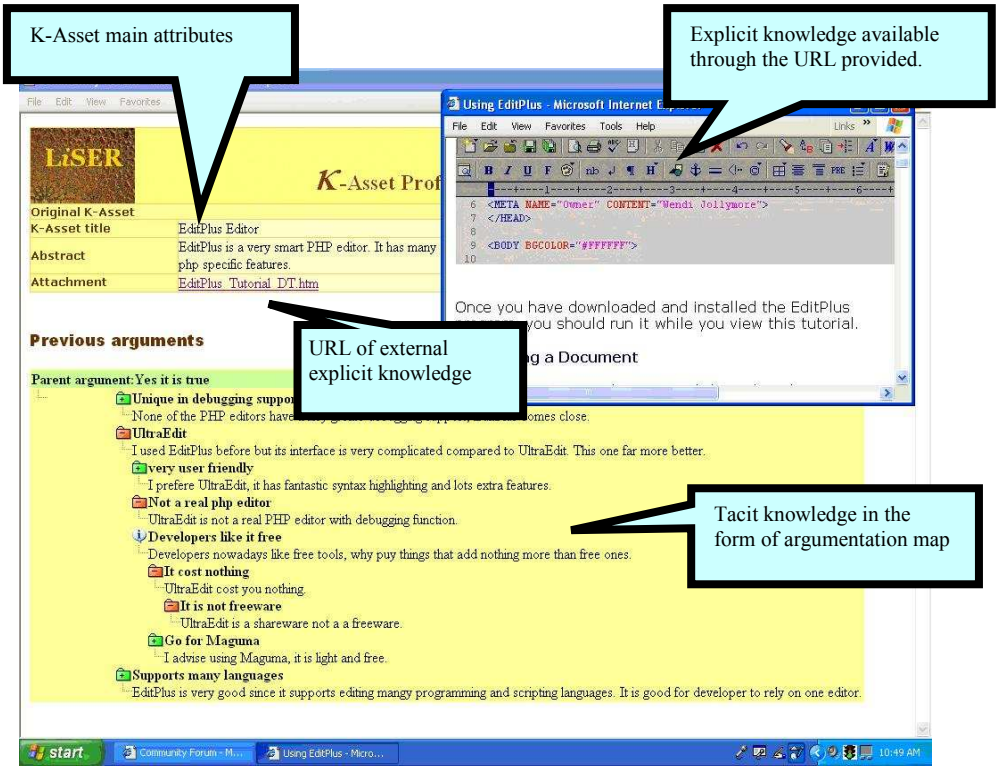

Fig. 3 An implementation of the proposed model in the form software-engineering kno management system

5 Conclusion

Knowledge is the winning factor for today's organisations and knowledge management systems constitutes the tool through which organisations know what they know and what they should know. Unfortunately, knowledge management systems were structured more on formally structured knowledge while less attention is paid to capturing tacit knowledge. Software engineering is a very knowledge-intensive domain and a great portion of software organisations' knowledge is realised in tacit form. It is usually held in professionals' heads as practical know-how that might be lost at any time. Therefore, there is an urgent need to capitalise tacit knowledge as well. This paper proposes a knowledge model that caters for capturing both tacit and explicit knowledge in the software-engineering domain. The model integrates both explicit knowledge in the form of software artefacts and tacit knowledge in the form of arguments that constitute the context behind the creation and validation criteria of captured knowledge. In order to maintain the validity of captured knowledge, a knowledge lifecycle is defined where the captured knowledge is continuously examined by peer knowledge workers, and the relevance of respective knowledge assets are determined accordingly.

References:

[1] Becker, K. (1997). Integrating Voting Techniques into a Discussion Rationale Model-based GDSS. *ISDSS'97*, Retrieved June 26 2002, from: http://inforge.unil.ch/isdss97/papers/71.htm

[2] Berlin, L.M., *et al.* (1993). Where did you put it? Issues in the design and use of a group memory. *Proceedings of the INTERCHI'93 Conference on Human Factors in Computer Systems*, 23-30, New York: ACM.

[3] Cummings, N. (1999). How OR conserves precious resources in the hire & fire age. *Operational Research Newsletter*, January 1999.

[4] Conklin, E. J. (1996). Designing Organisational Memory: Preserving Intellectual Assets in a Knowledge Economy. Retrieved May 2000, from : http://www.gdss.com/DOM.htm

[5] Schreiber, G., *et al*. (2000). *Knowledge engineering and management: the CommonKADS methodology*. Cambridge, Mass. : MIT Press, ISBN: 0262193000.

[6] Johannessen, J. A., Olaisen, J., Olsen, B. (2001). Mismanagement of tacit knowledge: the importance of tacit knowledge, the danger of information technology, and what to do about it, *International Journal of Information Management,* **21**, 3-20.

[7] Kokkoniemi, J. (2006). A Preliminary Model for Generating Experience Knowledge Based Artifacts. *Proceedings of the 39th Annual Hawaii International Conference on System Science (HICSS'06*, track 7, p.153b.

[8] Koskinen K., Pihlanto P., Vanharanta H. (2003). Tacit knowledge acquisition and sharing in a project work context. *International Journal of Project Management*, Volume 21, Number 4, May 2003, pp. 281-290(10), Elsevier Science.

[9] Lee, J. (1991). Extending the Potts and Bruns Model for Recording Design Rationale, IEEE.

[10] Liebowitz, J. (2001). *Knowledge Management: Learning from Knowledge Engineering*. pp57-62, ISBN 0-8493-1024-5, CRC Press, USA.

[11] MacLean, A., Young, R.M., Bellotti, V., Moran, T. (1991). Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interactio*n, 6, 3 & 4, pp. 201-250.

[12] Nonaka, I. (1998). The Knowledge Creating Company, *Harvard Business Review on Knowledge Management*, Harvard Business School Press.

[13] Rittle, H., Webber, M. (1973). Dilemmas in a general theory of planning. *policy science*.

[14] Tiwana, A. (2000). *Knowledge Management Toolkit, The: Practical Techniques for Building a Knowledge Management System*, , Prentice Hall PTR.

[15] Zack, M. H. (1999). Managing codified knowledge, *Sloan Management Review* 40 (4), p. 45-58.