Kevin C. Desouza

Barriers to Effective Use of Knowledge Management Systems in Software Engineering

An organization must recognize that information technology is only one means to foster knowledge.

Knowledge management has made headway in all fields in recent times and continues to emerge as a pivotal task for organizations to survive in today's competitive marketplace. The surge in interest can be attributed to the realization that organizations must not only exploit tangible resources but also focus on intangibles for effective and efficient attainment of organizational goals.

Researchers and practitioners have suggested a multitude of approaches to managing knowledge, most of which can be categorized broadly into codification and personalization approaches [1].

The codification approach focuses on amalgamating individual knowledge in organizations, putting it in a cohesive context and making it available to organizational members. Such an approach entails separation of knowledge from its creator.

The personalization approach is the opposite, in which knowledge sharing is fostered through people-to-people interactions and dialogue. In this approach, knowledge is not separated from its source, as one needs to identify the source of the required knowledge to request it. This is in contrast to the codification approach wherein a central repository of knowledge is provided. Many knowledge management initiatives rely on information technology as an important enabler. Use of databases and data warehouses as the central repositories for capturing and storing knowledge is common, and the use of email or group support systems is found to enable ubiquitous communication between members of an organization.

Software engineering is a highly knowledge-intensive domain, in which the keys to success are related to one's experience in one or more of the following: systems design, coding, testing, and implementation. Within each of these domains, there can be many more subdomains. For example, in coding, one can categorize expertise based on the different languages such as C, C++, and Java, or platforms like Unix and Solaris, among others. Due to the vastness of the field, seldom does one individual have all the resources to complete a project. Moreover, to keep pace with the developments in computer science, one has to constantly update oneself with the latest developments to prevent becoming obsolete.

Many IT and software firms have developed knowledge management systems in an effort to help programmers tap into their peers' experiences and to learn from each other. A vast majority of such systems employ the codification approach, in which a central repository holds knowledge under categories such as programming bugs, quality control reports, new developments, and so forth. Hence, if a programmer finds a bug, say, "+123942" while coding, he or she can look into the knowledge base and read about how one's peer solved the problem when encountered. This prevents reinvention of the wheel and makes programming a smoother experience.

While such knowledge-based systems are gaining popularity and have real benefits, they suffer from serious limitations. Here I outline three pivotal concerns among software engineers in using such systems. During the course of my research I spoke to 50 different engineers in 10 different organizations, spanning both large companies such as For-

PAUL WATSON

tune 100 organizations and mid-sized firms.

Three Concerns

In order of severity, what follows are key issues curtailing the effective use of knowledge management systems in software engineering.

Resistance to be known as an expert. A major concern of software engineers contrary to popular opinion is the fear of being known as an expert. Once titled an expert, software engineers find themselves being allocated to projects based on their past experiences, rather than those that may be more intellectually challenging and have room for learning. This fear was found to be persistent across all organizations, due to software engineers being reluctant to contribute to the knowledge bases and share their expertise.

Knowledge management systems store access statistics on individual records in the database. Thus one can see details such as how many times a particular document has been accessed, by whom, and how the person who accessed it assessed its relevance or quality, making it easy to deduce who the experts are and whose opinion matters.

One engineer said: "…if I contribute nuggets of know-how on how to run applications on the Unix box, soon I will be dubbed the 'Unix Guru' and that is all I will end up being in charge of… ."

Software engineering is a continuously evolving field, in which survival is dependent on keeping abreast with new developments and experimenting with the latest technologies. Hence, being stereotyped as an expert works to one's disadvantage and hampers rather than advances one's career.

Required knowledge cannot be captured and categorized. Software engineering knowledge is highly tacit in nature, much of which cannot be articulated well or be put in explicit format. Also, when one wants to contribute insights into a knowledge management system, the cost of doing so on average outweighs perceived benefits. For instance, a programmer at a large consulting firm in Chicago recalls one such experience: "…when I wanted to contribute a note to our knowledge systems, I first was frustrated trying to find the right category where to place it … my insight was not a bug report or a new development to report; all I wanted to do is explain how to tackle a particular runtime issue … I ended by inputting it in the miscellaneous category … and it took me over an hour to detail half a paragraph to capture all factors surrounding the issue… ."

Much of the knowledge in software engineering is highly contextual in nature, which calls for focused applicability. How a logical error was fixed in one scenario may have no bearing on another. Hence, to truly contribute insights one has to capture not only the new knowledge but also contextual factors in an explicit format, which is a costly endeavor.

Richness of alternative knowledge exchange mediums. The alternatives to using knowledge management systems posit several very inviting features. Take for instance, a programmer searching on how to fix bug "+E44222." When searching through the knowledge base, he or she is bound to find a description that is highly specific, such as "+E44222 is a runtime error, common reasons include… ." It is impossible to capture all circumstances under which the bug may occur or to even record all possible combinations of scenarios. Hence, the alternative of using the knowledge management system is to engage in dialogue with one's peers. A programmer can call one's peer to his or her desk, demonstrate the problem, and also learn first-hand how to solve it.

Nonaka and Takeuchi [2], in their seminal work on knowledge management in Japanese firms, call this process "socialization," and it is a key for exchange of tacit knowledge. Moreover, if one uses the knowledge system, he or she is bound to need physical contact with one's associates some time in the future; why not use this route from the beginning. When one engages in dialogue with his or her peers, knowledge exchanged surpasses that which is captured in information systems. A computer system can only capture items planned for and it is difficult to account for all scenarios at design time.

Remedies

It is not enough just to store knowledge in repositories; one has to reap intended benefits by exploiting the knowledge. While relying on information technology approaches should not be taken as the Holy Grail, they do

have pivotal value for knowledge management if used appropriately. First, an organization must allow for contribution to knowledge bases and encourage a knowledge-sharing culture by clearly defining incentives.

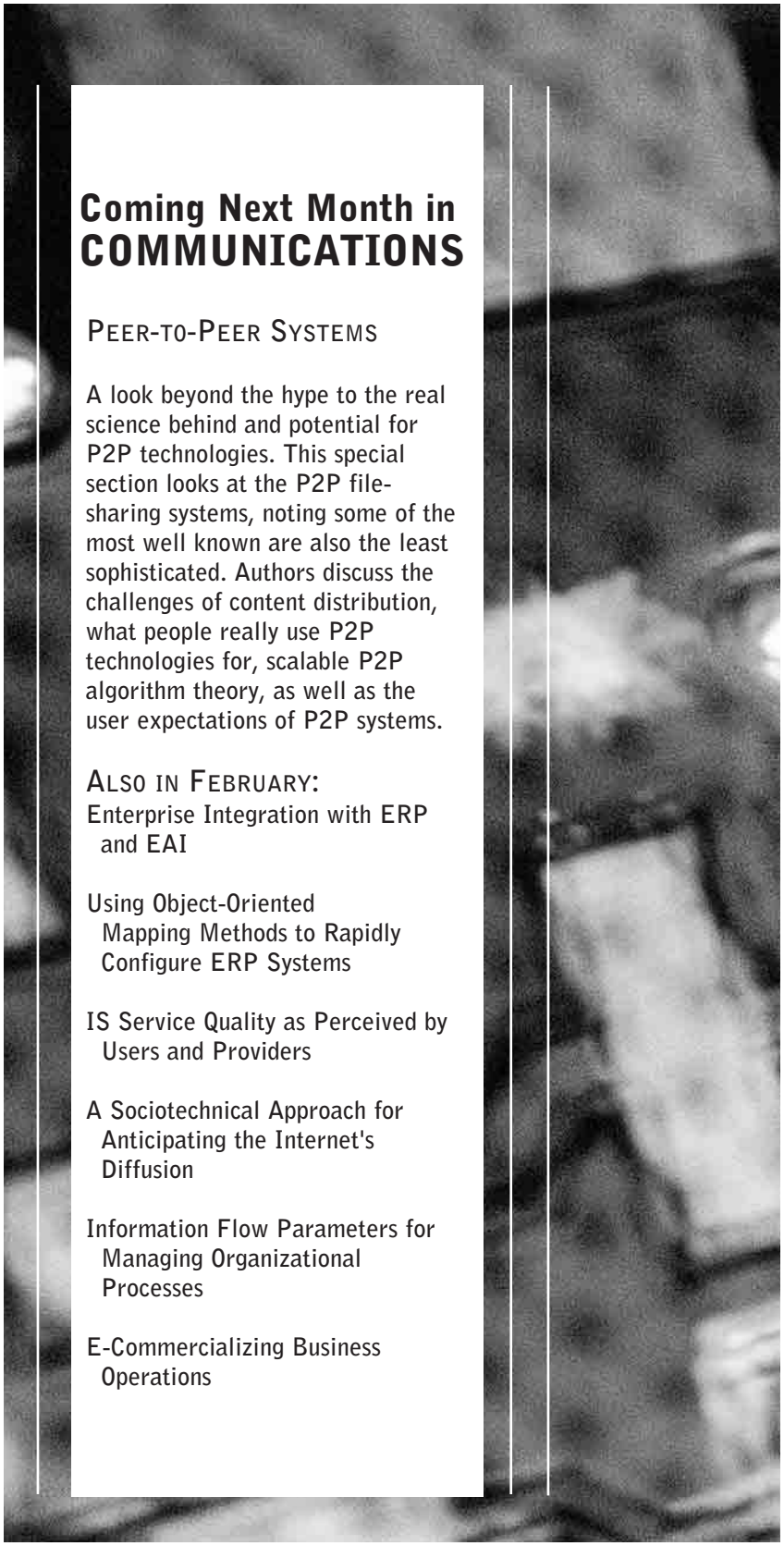
As I've noted, engineers do not want their expertise in a particular language or aspect of design to be the key determinant of stunted intellectual growth. Organizations must respect this and allocate people to projects not on what they know, but their potential to learn and explore. Second, knowledge management systems should encourage dialogue between individuals rather than just point to repositories. It is impossible to capture all expertise in databases. Hence technology must move away from this goal and foster communication.

Finally, an organization must recognize that information technology is only one means to foster knowledge and using metrics such as access to the knowledge base or number of postings may not be a true indicator of knowledge-sharing behavior of employees. C

References

1. Hansen, M.T., Nohria, N., and Tierney, T. What's your strategy for managing knowledge? *Harvard Bus. Rev., 77*, 2 (1999), 106–116.
2. Nonaka, I. and Takeuchi, H. *Knowledge Creating Company*. Oxford University Press, New York, 1995.

Kevin C. Desouza (kdesou1@uic.edu) is a research associate at the Center for Research in Information Management, the University of Illinois at Chicago.



Coming Next Month in COMMUNICATIONS

Peer-to-Peer Systems

A look beyond the hype to the real science behind and potential for P2P technologies. This special section looks at the P2P file-sharing systems, noting some of the most well known are also the least sophisticated. Authors discuss the challenges of content distribution, what people really use P2P technologies for, scalable P2P algorithm theory, as well as the user expectations of P2P systems.

Also in February:

Enterprise Integration with ERP and EAI

Using Object-Oriented Mapping Methods to Rapidly Configure ERP Systems

IS Service Quality as Perceived by Users and Providers

A Sociotechnical Approach for Anticipating the Internet's Diffusion

Information Flow Parameters for Managing Organizational Processes

E-Commercializing Business Operations