

Toward a Practical Solution for Capturing Knowledge for Software Projects

Seija Komi-Sirviö and Annukka Mäntyniemi, *VTT Electronics*

Veikko Seppänen, *University of Oulu*

Rarely has a professional field evolved as quickly as software development. Software organizations are continuously struggling to keep abreast of new technologies frequently changing customer requirements; and increasingly complex software architectures, methods, and tools. Recently, many organizations have come to understand that to succeed in the future, they must manage and use knowledge more effectively at individual, team, and organizational levels.^{1,2} Efficient creation, distribution, and reuse of up-to-date knowledge are critical success factors that unfortunately remain difficult to achieve in practice.³⁻⁵

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

So, how can the existing knowledge in a software organization be captured efficiently? To try to answer this question, we conducted a case study in an independent business unit of a global corporation developing software-intensive electronic products. The company wanted to improve the capture and reuse of

software development knowledge for a particular project. It had made several attempts to improve knowledge reuse, but all these attempts had failed. Why did the earlier attempts not succeed? What would be a working solution? We set out to study those questions to learn from the previous difficulties and build on the previous successes.

Analyzing the status of KM-based SPI

Employee interviews and relevant documents revealed that both the managers and designers felt that a lot of knowledge was being wasted. Existing knowledge was difficult to find, and when found it was not reusable. The practices implemented earlier had obviously not been successful.

The underlying goal had been to reduce software defects by increasing the knowledge

A needs-based approach to reusing software development-related knowledge can overcome past failures at knowledge reuse.

transfer between different projects. The information to be shared was stored in a *Lessons to Learn* database. Interviews clearly indicated that many project managers were not aware of the database and few used it. An analysis of the database revealed that a number of entries were incomplete and only one of the four thematic sections was in active use. According to the database concept owner, this was because preparing database entries was time-consuming and administering the data was difficult. Moreover, the data's accuracy and relevancy were not obvious, because most of the data was provided without structure.

Another way to share knowledge between projects was *Data Transfer Days*. These meetings focused on analyzing past problems and success stories. The participants captured and shared important knowledge during the meetings, even though they had trouble remembering past successes and pitfalls once their projects had ended. The intention was to analyze, package, and save the results of these meetings as a reference for new projects. Unfortunately, the enthusiasm usually disappeared at this point. The meetings were useful mainly for those who were able to attend. Nevertheless, free face-to-face conversation between group members turned out to be a better way of sharing knowledge than the database (compare this to Thomas Davenport and Laurence Prusak's idea that the human network is a highly efficient knowledge-sharing mechanism¹).

As you can see, neither Data Transfer Days nor, particularly, the Lessons to Learn Database were working as initially intended. An obvious reason for the latter was that the project management processes did not incorporate guidelines for the storing or searching of knowledge. Efficient use of the database would have required more disciplined processes and much more effort at capturing, packaging, searching, maintaining, and reusing the knowledge. Furthermore, most project managers were too busy coping with their everyday problems and were unwilling to undertake any further duties. Our interviews also indicated that software designers tended to trust anyone nearby, rather than any specific experts or the shared database.

Researchers who have investigated these problems elsewhere have found problems similar to ours and give these reasons for reuse failures: the knowledge capturing

process is too informal, is not incorporated into the engineering processes, or is not supported by the structures of the organization.⁶ Davenport and Prusak have stated that if you start with technology-centered solutions (for example, a database) and ignore behavioral, cultural, and organizational change, the expected advantages never materialize.¹

Looking for a new solution

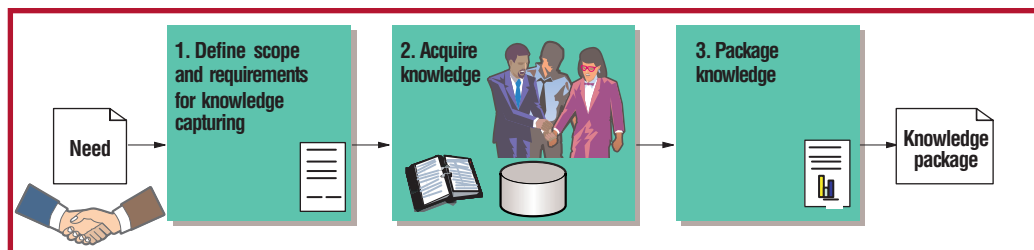
The organization set a challenging requirement: new solutions should have minimal impact on the software development organization and processes and should not require new technologies. Because the existing processes should not be touched, simple, manual, and offline means were preferable, removing the excessive burden of KM. This SPI action aimed to create a process that would help the company acquire experience from existing sources—such as the company's databases and individuals—that it could apply to ongoing SE projects. One new idea to define and test was to use the SPI experts as knowledge-capturing agents instead of having software developers do it by themselves, on the fly. This method viewed projects as individual customers that required specific knowledge. Efforts would focus on the customer's current knowledge requirements, as opposed to a large-scale acquisition, analysis, packaging, sharing, and updating of knowledge for subsequent projects.

The company's SPI persons and external experts established a new approach for capturing knowledge. This new approach consisted of a *knowledge-capturing project* and *customer projects*. The former gathered knowledge from relevant sources and packaged and provided it to a customer project for reuse on demand. This solution neither changed the organizational setting nor required any new tools. The knowledge would come from existing sources such as project final reports, error databases, discussion forums, and—most important—people. Later, this assumption proved true. Figure 1 shows a simplified capturing process.

Unlike other approaches,^{6,7} this one did not expect ongoing software projects to supply their experience. The knowledge-capturing project is similar to the analysis organization in the Experience Factory framework⁶ in that it analyzes knowledge and packages it into reusable assets. However, it does this for the

The underlying goal had been to reduce software defects by increasing the knowledge transfer between different projects.

Figure 1. The knowledge-capturing process.




customer projects' immediate needs.

Together with the company, we tested this approach on a project that urgently needed interface-related knowledge. This special knowledge was spread among various documents, memos, databases, and people. The Lessons to Learn Database did not provide this knowledge; Data Transfer Days hadn't helped. The customer project's needs were structured to indicate what specific knowledge was required, what form of knowledge would be reused. The needs were also divided into process- and product-related knowledge. The former included, for example, software design and testing tasks, roles, organizations, skills, methods, and tools. The latter included descriptions and interfaces of products and product family hierarchies.

As planned, we followed the knowledge-capturing process, using semi-structured interviews as the main technique to acquire knowledge. The process took 300 hours; the most laborious phase was experience packaging, which took more than one-half of the

time. The delivered interface knowledge package fully met all its requirements: the customer project retrieved needed knowledge of the existing interfaces. The selected approach worked well, and the customer project was served the required knowledge just in time.

Although we acknowledge the limitations of a single case study, we do not hesitate to call into question technology-centered solutions as the main means for managing software development knowledge. We feel our study is a first step toward a more comprehensive needs-based KM approach. We will continue to expand the use of the just in time KM process and will work to make it a part of normal project initiation procedures. Over time our efforts should help provide structured and packaged information that will have real value for software projects. 

References

1. T.H. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard College Business School Press, Boston, 1998, p. 199.
2. I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford Univ. Press, New York, 1995, p. 284.
3. V. Basili et al., "Implementing the Experience Factory Concepts as a Set of Experience Bases," *Proc. 13th Int'l Conf. Software Eng. and Knowledge Eng. (SEKE 01)*, Knowledge Systems Inst., Skokie, Ill., 2001, pp. 102–109.
4. T. Kucza et al., "Utilizing Knowledge Management in Software Process Improvement: The Creation of a Knowledge Management Process Model," *Proc. 7th Int'l Conf. Concurrent Enterprising (ICE 2001)*, Univ. of Nottingham, Center for Concurrent Enterprising, Nottingham, UK, 2001, pp. 241–249.
5. K. Schneider, "Experience Magnets: Attracting Experiences, Not Just Storing Them," *Proc. 3rd Int'l Conf. Product Focused Software Process Improvement (PROFES 2001)*, Lecture Notes in Computer Science, no. 2188, Springer-Verlag, Heidelberg, Germany, 2001, pp. 126–140.
6. V. Basili, "The Experience Factory," *Encyclopedia of Software Eng.*, vol. 1, John Wiley & Sons, New York, 1994, pp. 469–476.
7. A. Birk and C. Taus, "Knowledge Management of Software Engineering: Lessons Learned," *Proc. 10th Conf. Software Eng. and Knowledge Eng. (SEKE 98)*, Knowledge Systems Inst., Skokie, Ill., 1998, pp. 24–31.

For more information on this or any other computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

About the Authors



Seija Komi-Sirviö is a research scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland. She is visiting from VTT Electronics, Finland, where she has carried out research into software process improvement and metrics in applied research projects from 1994 onwards. As a research group manager, she has been responsible for initiating and managing both applied research projects and industrial development projects for a broad range of clients in software engineering. Her current research interests include software process and product improvement, measurement, and knowledge management. She received her MSc in information processing science from the University of Oulu, Finland. She is a member of the IEEE Computer Society. Contact her at ssirvio@fc-md.umd.edu.

Annukka Mäntyniemi is a research scientist at VTT Electronics, Oulu, where she has worked since 1998. She received her Master's degree in Information Processing Science from University of Oulu, Finland in 2001. Her thesis concerned the reuse of software development experiences. Her current research interests involve utilizing knowledge management in software process improvement and software reuse. Contact at Annukka.Mantyniemi@vtt.fi.



Veikko Seppänen is a software business research professor at the University of Oulu, Finland with almost 20 year's experience in software research and development. He finished his engineering doctoral thesis on software reuse in 1990 and his second dissertation on economic sciences in 2000. Seppänen has published about a hundred scientific and practical publications. He was an Asla Fulbright scholar at UC Irvine in 1986–87 and a JSPS Postdoctoral Fellow at Kyoto University in 1991–93. His present research involves software business strategies and models, including value network based approaches to industrial marketing, acquisition and use of commercial-off-the-shelf software components and products, and knowledge-driven software product and business development methods. Contact him at veikko.seppanen@oulu.fi.