Improving Knowledge Sharing in Distributed Teams by Capturing and Recommending Informal Knowledge[1]

Hans-Jörg Happel

FZI Forschungszentrum Informatik
Karlsruhe, Germany
happel@fzi.de

Walid Maalej

Technische Universität München
Munich, Germany
maalejw@in.tum.de

Due to an increasingly distributed workforce, teams are often separated by organizational, geographical or temporal boundaries, which cause substantial collaboration and knowledge exchange problems. Accordingly, tool-support for collaboration and knowledge sharing is particularly important for distributed teams. In this paper, we discuss how knowledge exchange in the domain of software development can be improved by capturing informal knowledge and recommending to share or access it.

1 Introduction

The development and maintenance of large and complex systems, which exceed the abilities of a single human worker, led to the concept of division of labor: decomposing a system into smaller modules creates manageable units of work. Decomposition introduces dependencies among the modules and thus requires coordination to create an integrated system in the end. Since an increasing number of projects are carried out in a distributed fashion, an effective and efficient coordination and knowledge sharing is considered a major success factor for such organizations [Cum04, Han99].
In this paper, we introduce our position on how to improve knowledge sharing in distributed teams. First we characterize coordination and knowledge sharing in distributed work settings. Then we discuss means for supporting knowledge sharing in a collaborative software development environment.

2 Coordination and knowledge sharing in distributed teams

Distributed work has become popular in recent years due to several reasons such as cost reduction, availability of human resources or intra-organizational collaboration [Ols00]. The modularization of complex artifacts is thus often mirrored by a modularization of the project teams, which produce these artifacts [KM06].

[1] This work has been partly supported by the TEAM project, which is funded the EU-IST program under grant FP6-35111 and the BMBF-funded project WAVES.

Many distributed and outsourced projects suffer from coordination problems since the overall coordination capacity is lower due to a reduced communication bandwidth [Ols00]. Since distributed teams are less efficient than collocated ones, collaboration research is focusing on tool-support for creating “virtual 30 meters” [HM03]. Core challenges in realizing these virtual 30 meters are coordination and knowledge sharing among the team members.

The need for coordination stems from the modularity of the artifacts under development. The decomposition of artifacts creates dependencies, which require coordination. Coordination can thus be defined as “the process of managing dependencies between activities” [MC94]. Accordingly, most coordination requirements can be traced back to explicit dependencies among the subsystems of artifacts. Coordination issues are central to collaboration research, which e.g. analyzes suitable coordination mechanisms for different kinds of dependencies [Tho67] and tools for supporting communication, awareness and workflow management.

While coordination problems in distributed settings are understood and supported in a considerable way, we argue that further support for knowledge sharing is required. Knowledge sharing can be defined as the ”dual problem of searching for (looking for and identifying) and transferring (moving and incorporating) knowledge across organizational subunits” [Han99].

In contrast to coordination, knowledge sharing is not directly rooted in explicit dependencies of technical artifacts, but in dependencies among organizational entities. The modularity of the organizational subsystem has a deep impact on the communication patterns of an organization since it forms channels and filters along organizational interfaces, to reduce complexity by selecting relevant information [HC90]. However, this has negative impact on knowledge sharing across organizational units, since the average information flow runs dry with increasing organizational or geographical distance. Additionally, distributed work settings amplify barriers for knowledge sharing such as reduced motivation and trust [CC02, Des03]. The allocation of knowledge can thus be inefficient when collaboratively developing a system in a distributed organization.

3 Supporting knowledge sharing in distributed software teams

Due to the popularity of Open Source and offshoring development models, software development is one of the most distributed and knowledge intensive businesses, in which people with different backgrounds and expertise levels collaborate. The crucial impact of coordination on the efficiency of software development projects is well-known [Ols00] and empirical studies have shown that technically driven design decisions influence coordination and knowledge sharing in development teams, which can in turn decrease productivity [HM03].

While collaboration support gets increasingly integrated into software development tools [CS+04], the exchange of knowledge and expertise in development teams remains largely unsupported beyond basic communication and document sharing tools. Most work on knowledge support is focused on supporting knowledge access. Examples are systems like Hipikat [CSB05] that help developers to access reusable know-

ledge artifacts such as documents or source code. However, these approaches assume an existing repository of knowledge artifacts and do not discuss how these repositories are built and maintained. The Experience Factory [BCR94] (EF) describes a framework for building repositories of reusable knowledge. However, an EF requires costly knowledge extraction and refinement processes to create high-quality reusable assets, as well as centralized organizational responsibility in order to maintain the knowledge repository.
Finally, the mentioned approaches fail to support sharing of small pieces of experience, which are usually exchanged during informal communication in collocated settings. For example, during a coffee break, team members share activities they have carried out to resolve a certain bug, lessons learned on configuring a particular component before reusing it, the name of an expert on a particular technology, or a knowledge source such as a Web site to answer development questions.
A notable approach in the direction of sharing such small pieces of informal knowledge is the mylyn project [KM06], which monitors interactions inside an IDE, while developers carry out their tasks. Based on these interactions, mylyn computes a relevance value for artifacts in a given task and hides or blurs them in the user interface when they are deemed not relevant for the current task. This approach is called "Task-Focused User Interface". Mylyn is implemented as a plug-in for the Eclipse Integrated Development Environment (IDE) and allows sharing the gathered information with other developers.

We envision an advanced approach to support knowledge sharing in distributed software teams. It is based on the following assumptions:

- A lot of useful information and pieces of experiences are implicit and not formally captured by the developers, in form of documentation.
- Developers have private explicit knowledge, which is not shared with collaborators (especially in distributed work settings).

We aim to support both the capturing of implicit knowledge and the diffusion of existing private knowledge in distributed software development teams. Basic enablers for our approach are the observation of developer's activities and the exchange of such data among the developers in a team. Observation leads in particular to sessionizing the developer's workday into problem-solution cycles, recognizing problem situations (such as information needs or program errors) and capturing information accessed to solve such problems. In our prototypical implementation, this is realized by a context observation framework, which records developer's interactions with his tools such as the IDE, the web browser and the collaboration and communication tools. These interactions are further processed, aggregated and then persisted in an ontology-based, distributed metadata infrastructure (c.f. [MH08]).
Regarding the diffusion of existing knowledge, we aim to incentivize developers to share knowledge, by recommending not only the information to share but also other team members to whom such information is valuable. Therefore, our approach analyzes the overall information behavior within the working group, mainly based on query logs (c.f. [Hap08]). Query logs are also used to identify sought information, which is not yet available for a team. Complemented by context observation data, we

are able to find out developers with expertise on a certain topic. We then recommend them to capture particular information. As an example, developers fixing a bug will be asked to document their solution, if the system knows that other collaborating developers faced the same or similar issues.

4 Conclusion

In this paper, we described the problem of coordination and knowledge sharing in distributed teams, and argued that existing work focuses on coordination, while knowledge sharing is still neglected – especially in software development. Based on this, we discussed existing work in collaborative software development and sketched our approach for fostering knowledge sharing and capture. Capturing and formalizing small pieces of developers' experiences, as well as recommending to sharing private information to other colleagues who need it are the mail enablers. We are currently working on the implementation and evaluation on these concepts in several projects.

5 Literature

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. Experience Factory. In John J. Marciniak (ed.), Encyclopedia of Software Engineering, John Wiley & Sons, 1994.

[CC02] Angel Cabrera and Elizabeth F. Cabrera. Knowledge-sharing dilemmas. Organization Studies, 23:687–710, 2002.

[CD+04] Li-Te Cheng, Cleidson R.B. de Souza, Susanne Hupfer, John Patterson, and Steven Ross. Building collaboration into IDEs. Queue, 1(9):40–50, 2004.

[CSB05] Davor Cubranic, Janice Singer, and Kellogg S. Booth. Hipikat: A project memory for software development. IEEE Trans. Softw. Eng., 31(6):446–465, 2005.

[Cum04] Jonathan N. Cummings. Work groups, structural diversity, and knowledge sharing. Management Science, 50(3):352–364, 2004.

[Des03] Kevin C. Desouza. Barriers to effective use of knowledge management systems in software engineering. Commun. ACM, 46(1):99–101, 2003.

[Han99] Hansen, M.T.: The search-transfer problem: The role of weak ties in sharing knowledge across organization subunits. Administrative Science Quarterly, 44:82–111, 1999.

[Hap08] Hans-Jörg Happel: Closing information gaps with inverse search. In 7th International Conference on Practical Aspects of Knowledge Management, LNCS. Springer, 2008.

[HC90] Henderson, Rebecca M.; Clark, Kim B.: Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. In: Administrative Science Quarterly 35 (1990), p. 9-30.

[HM03] Herbsleb, James D. ; Mockus, Audris: Formulation and preliminary test of an empirical theory of coordination in software engineering. In: Proceedings of the 9th European software engineering conference. ACM Press, 2003, p. 138-137

[KM06] Mik Kersten and Gail C. Murphy. Using task context to improve programmer productivity. In SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering, p. 1–11, NY, USA, 2006. ACM.

[KS95] Kraut, Robert E. ; Streeter, Lynn A.: Coordination in Software Development. In: Commun. ACM 38 (1995), March, No. 3, p. 69-81

[MH08] Walid Maalej, Hans-Jörg Happel: A Lightweight Approach for Knowledge Sharing in Distributed Software Teams. PAKM 2008, LNCS. Springer, 2008

[MC94] Malone, T. W. & Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys, 1994 (March), 26 (1), 87-119.

[Ols00] Gary M. Olson and Judith S. Olson. Distance matters. Human-Computer Interaction, 15(2/3):139–178, 2000.

[Tho67] Thompson, James D.: Organizations in Action. McGraw-Hill, New York, 1967