Towards Patterns to Enhance the Communication in Distributed Software Development Environments

Ernst Oberortner, Irwin Kwan, Daniela Damian
e.oberortner@gmail.com, irwink@cs.uvic.ca, danielad@cs.uvic.ca

Distributed Software Development (DSD) is an emerging research area in software engineering. Several conducted research studies identified similar communication problems among DSD teams and tried to solve them. In this paper we present patterns that we have identified while surveying state of the art research studies. The patterns can help to organize DSD teams better in order to enhance their communication. We also highlight some potential future research challenges.

1. Introduction

Nowadays, several development teams of large software houses are distributed globally [HRG+08,MHG+10,Lan09]. The collaboration among distributed software development (DSD) teams is based upon the team members' communication. The better the communication, the better the collaboration, and the better the success of the resultant software. DSD teams communicate about various aspects of the intended software. For example, the software's requirements must be defined, updated, and clarified. Technical utilities, such as mailing lists, online forums, software repositories, or bug tracking systems, facilitate the communication during the whole software development process.

The literature consists of many studies and tools to enhance the communication of DSD teams, such as [DKM10,HM06,SR07]. Many studies use data mining techniques [HK05] and utilize social network analysis methods [WF94] to discover the communication structures of DSD teams. Mining the data of communication and software repositories can help, for example, to leverage invisible relations between globally distributed team members. Another example is to support newcomers, helping them to collaborate with task-related experienced team members in order to become productive more quickly.

We present patterns to improve the communication within DSD teams. The patterns describe (1) how to organize co-located teams to enhance the communication with remote teams and (2) how to support the DSD team members to find other DSD team members with some wanted professional skills.

Agile software development is a software development process that is mostly utilized in DSD teams [Eck10]. In this paper, we do not focus on the software development process that DSD teams utilize. However, the communication during the whole software development process is an important aspect. The

patterns presented in this paper can improve the communication independent of the utilized development process.

This paper is structured as follows: In Section 2 we justify the importance of the communication within DSD teams with an example. Section 3 explains the pattern's common terminology. Then, in Section 4 we describe the pattern's format. The main contribution of this paper is in Section 5, the patterns, and we also discuss the relationship between them. In Section 6 we highlight some potential future research challenges for DSD environments. The paper concludes with Section 7.

2. Motivating Example

In this section we justify the importance of the presented patterns to enhance the communication within DSD teams. Imagine a DSD development team, as exemplified in Figure 1, that is globally organized. The illustrated DSD team consists of one software designer and three software developers.

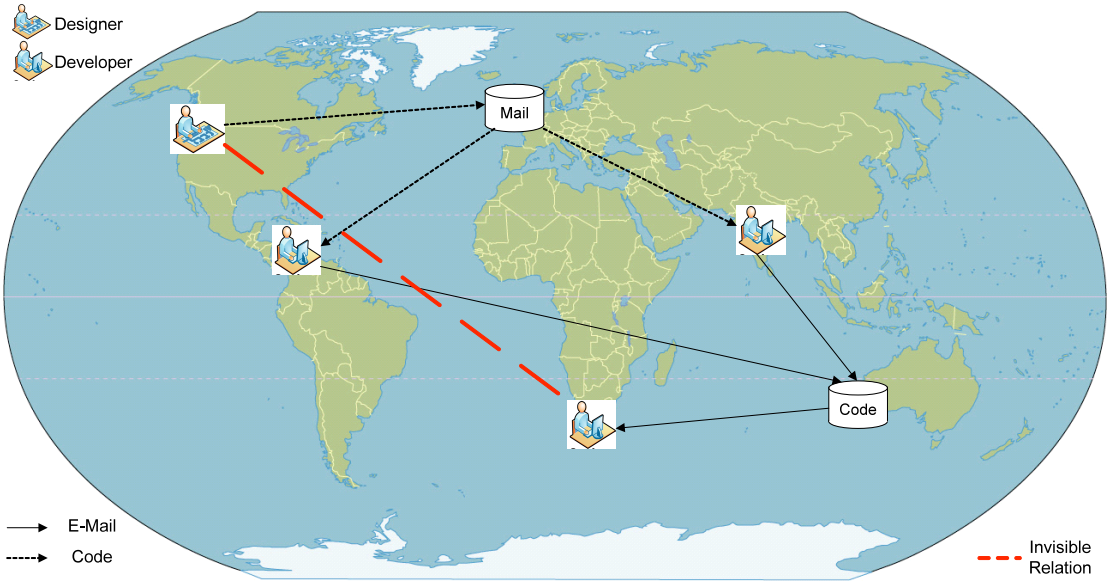

Figure 1 A motivating example

For example, a software designer, located in North America, propagates design updates to, in his opinion, related software developers via e-mail. The developers implement the required software updates and submit the new code into the source code repository. All developers receive a notification about the updated code, enabled by a special feature of the source code repository.

As illustrated, the software designer in North America does not directly communicate with the one developer located in South Africa. The developer in South Africa recognizes software updates through the recently submitted code into the software repository. It can happen that the new code is inconsistent with the code of the developer in South Africa. Such situations occur often in DSD teams because there exist invisible dependencies between the DSD team members.

In the illustrated scenario, mining only the mailing list or just the source code repository is not sufficient because the invisible dependency cannot be uncovered from just mining the mail repository. Only a combination of mining the mail and the source code repository's data identifies the invisible dependency. Detecting invisible dependencies can enhance future communications between the globally distributed software designer and the developers.

3. The Patterns' Terminology

The survey's studies use various terms that have a different naming but common meaning among all studies. In this section, we explain the survey's common terms in order to improve the understandability and to avoid misunderstandings.

The term **distributed software development (DSD)** elates to a software development strategy where the team members are distributed on various geographical locations. The team members are located at various **sites**. As illustrated in the motivating example (see Section 2), the designer is located in North America, whereas the developers are located in Asia, South Africa, and South America. In this case, North America, Asia, South Africa, and South America are the sites of the DSD team. Usually, each side consists of a group of team members. We term team members that reside on the same side as **co-located** team members, whereas team members that reside on different sites are **remote** team members.

All team members must **collaborate** with each other during the whole software development process in order to gather the requirements correctly, to design, develop, and test the software. The DSD team members use existing technical **communication** facilities to collaborate with each other, such as mailing lists, software repositories, bug tracking systems, or instant messengers. Such communication facilities can be used, for example, to coordinate task-related activities between the DSD team members and to propagate information to appropriate DSD team members. Similar to Damian et al. [DIS+07], we base the definition of **awareness** on Dourish and Bellotti [DB92]: Awareness is "*an understanding of the activities of others, which provides a context for one's own activities*".

A **social network** gives information about who communicates with whom in the DSD team. A social network can differ from the organization structure of the DSD team and can give additional information about the social interactions between the DSD team members.

4. The Patterns' Format

In the Table 1 we explain the pattern format that we use throughout the paper. Our pattern format is a subset of to the pattern format utilized in the GoF book [GHJ+95].

Problem:	We utilize a driving question in order to explain the pattern's problem.
Forces:	Regarding the Language of Shepherding [Har99], forces drive the problem. In our pattern format, forces explain and motivate the problem in more detail. We confirm the problem with papers discovered during a literature review.
Solution:	In the solution section we explain the pattern's solution to the problem. We also try to visualize to pattern's solution graphically.
Consequences and Resulting Questions:	A pattern's consequences are the side effects and trade-offs of applying the pattern. Consequences describe arising questions in case of utilizing the pattern.
Related Patterns:	In this section we list similar patterns that we have discovered in the literature.
Known Uses:	In the Known Uses section we list several studies that utilize the pattern or that have discovered the importance of the pattern.

Table 1 The utilized pattern format

Following Alexander [Ale77], we annotate each pattern with one or two asterisks. One asterisk denotes that the pattern is subject to further investigations. Based on our observations and findings during the literature survey, patterns annotated with two asterisks are valid.

5. Patterns to Improve the Communication

This section covers the main contributions of this proceeding. We present three patterns that cover various aspects of a DSD team's communication.

5.1 Pattern: *COMMUNICATION BROKER* **

How to enhance the communication between remote teams?

In DSD teams, challenges like miscommunication, misunderstandings, and how to share information are likely more severe than in co-located teams because of communication problems [HM06]. Also, in DSD environments the communication between co-located team members is more efficient than between remote team members. DSD teams are volatile and hence, the team members change frequently [DKM10].

Identify within each co-located team one *COMMUNICATION BROKER* that is responsible for the communication with the remote teams.

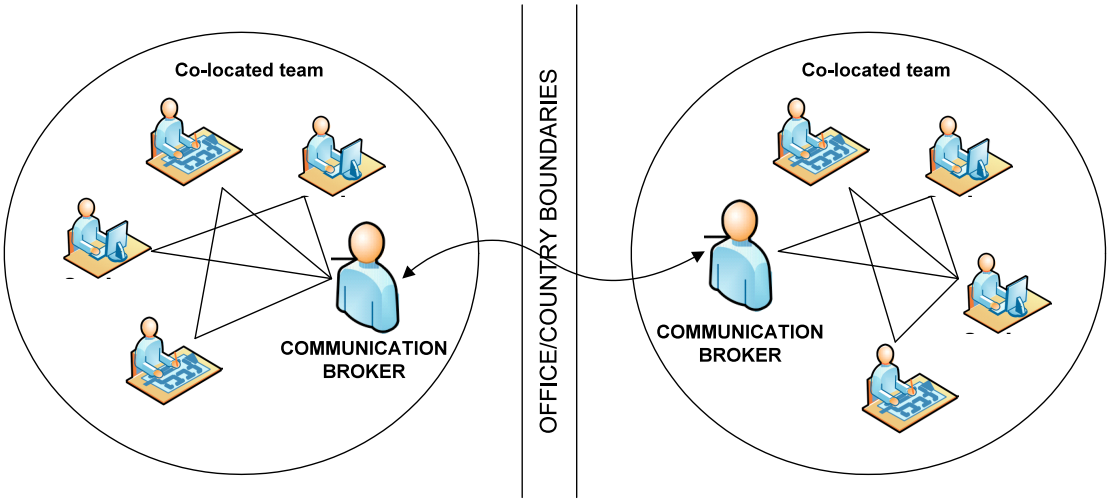

Figure 2 The *COMMUNICATION BROKER* pattern

In Figure 2 we visualize the *COMMUNICATION BROKER* pattern graphically. Within each co-located DSD team, one team member must be identified that is responsible for the communication with the remote teams and their team members. The *COMMUNICATION BROKER* observes the communication between its' co-located team members and propagates important issues, such as updates, changes, or extensions of the software, to the remote teams and their *COMMUNICATION BROKER*. The *COMMUNICATION BROKER* receives the important issues and propagates them to its' co-located team members.

One benefit of identifying *COMMUNICATION BROKERS* is that within each co-located team there is one central switching point that has knowledge about the communication within its' co-located team. The *COMMUNICATION BROKER* is mostly personally known within each co-located team, resulting in a trusted relationship between the *COMMUNICATION BROKER* and its' co-located team members. Because the *COMMUNICATION BROKER* communicates with remotely located *COMMUNICATION BROKERS*, a personal relationship evolves that can result in a trusted communication between the *COMMUNICATION BROKERS*. Because of the personal and trusted relationships between the *COMMUNICATION BROKERS* the communication can get enhanced.

COMMUNICATION BROKERS can have an information overflow of the communicated issues within and across co-located team members. Furthermore, the question about how to find appropriate *COMMUNICATION BROKERS* must be answered. This question relates to the *DISCOVERING THE EXPERTS* patterns (see Section 5.2). A further relevant question concentrates on the dynamic of DSD teams [DKM10]. What happens if the *COMMUNICATION BROKER* leaves the DSD team or the organization?

Related Patterns:

- Scott et al. introduce the Façade pattern that is similar to the *COMMUNICATION BROKER* pattern [SIG+05].

- Each small team of should have a Scrum Master [RJ02] of an agile software development process is comparable to the *COMMUNICATION BROKER* pattern. A Scrum Master is responsible for coordinating the team and the communication within the team. But, a Scrum Master is not a team leader.

Known Uses:

- Wolf et al. follow a three step filtering process to build a task-based social network [WSD+09]. The authors use the filtering process to identify *COMMUNICATION BROKER* in order to discover which team members contributed to a build failure.
- During a conducted web-based survey within R&D DSD teams, Hinds and McGrath [HM06] identified that a *COMMUNICATION BROKER* is necessary.

5.2 Pattern: *AWARENESS NETWORK**

How to find DSD team members with a required expertise and professional skills?

At least 70% of development time is spent on communication [DT87]. Finding appropriate experts in a globally DSD team to communicate with to gather additional experience, is a problem [MH02]. Team members who recently joined the DSD team are not aware of the other team members' professional background, skills, activities and tasks [EC06]. The network of assigned tasks differs from the social interactions within the DSD team [DIS+07]. To become productive more quickly, newcomers must be aware of whom to contact in case they have some task-specific questions [SR07].

Construct an *AWARENESS NETWORK*, i.e., a social network mined from several project repositories, which reflects the DSD team members' communication content.

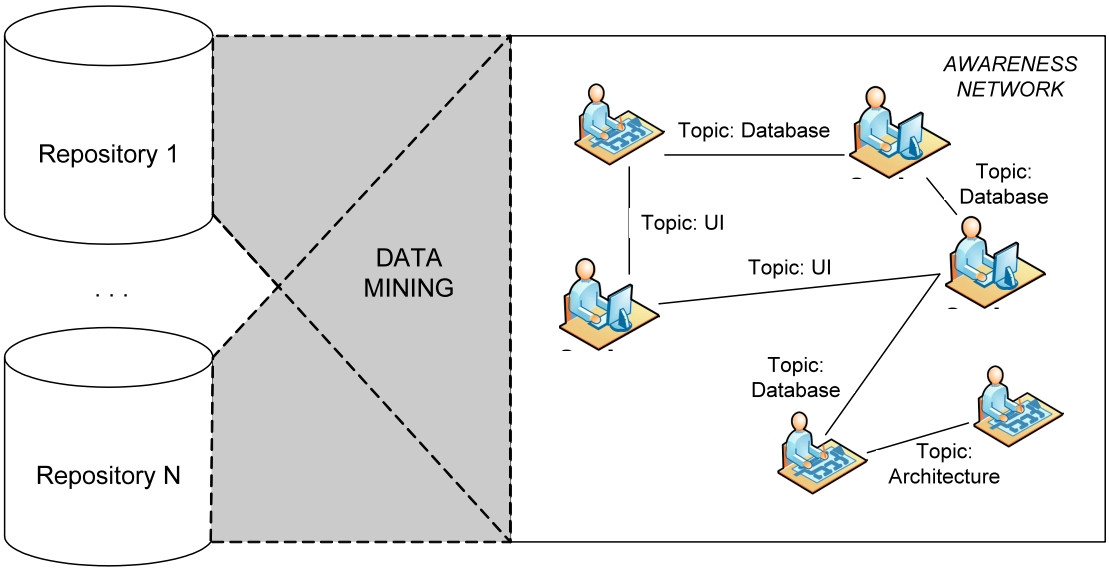

Figure 3 Constructing an AWARENESS NETWORK

In Figure 1 we sketch how to construct an AWARENESS NETWORK by mining multiple repositories, such as mailing lists, source code repositories, or bug tracking repositories. It is particularly important to gather the topic of the DSD team members' interactions in order to discover the team members' professional skills and in which interactions they participated. For example, if a developer that submits frequently to the source code repository new code for the database access and participates frequently in database topics in the mailing lists, then it is highly possible that this developer is an expert in database development. The resultant AWARENESS NETWORK should be accessible to all DSD team members, making it possible to search for team members with a desired expertise.

An *AWARENESS NETWORK* can speed up the communication within the DSD team. Newcomers can increase their productivity because they immediately know whom to contact regarding which questions. Invisible relationships can be detected because the *AWARENESS NETWORK* differs from the network of assigned tasks to the DSD team members.

Mining the data of multiple repositories is necessary because mining just one repository can lead to an inaccurate *AWARENESS NETWORK* [HWC06]. Mining historical data can limit the *AWARENESS NETWORK*'s accuracy. Therefore, an accurate and current status of work must be provided [DIS+07]. A permanent update of the *AWARENESS NETWORK* is needed, but keeping the *AWARENESS NETWORK* up-to-date is challenging because DSD teams are volatile and the team members change their roles or leave the team frequently [SR07]. Redmiles and deSouza [SR07] have discovered three factors that influence the awareness network: (1) the organizational reuse program, (2) the software developers' experience, and (3) the software architecture.

Known Uses:

- Hossain et al. [HWC06] mined the e-mail repository and performed a text-based keyword finding to construct the *AWARENESS NETWORK*. A limitation is that mining one repository is not sufficient. Also the authors state that a context-specific search would result in a more accurate *AWARENESS NETWORK*.
- The Expertise Browser [MH02] is a web-based tool to support DSD team members finding experts to talk to. The tool mines a change management system and visualizes project-related expertise information.
- The Hipikat Tool [CM03] mines some project's source and mail repositories to support newcomers to find relevant solutions for their tasks.

5.3 Relationships between the Patterns

In DSD teams, many challenges and problems exist that usually do not arise in co-located software development teams. In this section, we discuss the patterns and explain their relationships.

There exist a relationship between the COMMUNICATION BROKER and the AWARENESS NETWORK pattern. The observations by Hossain et al. [HWC06] are in accord with Conway's Law [Con68]. In hierarchically organized companies it means that the higher a team member is in the organization's hierarchy, the higher the possibility to select this team member as the *COMMUNICATION BROKER*. To avoid an information overload of the *COMMUNICATION BROKER*, it is important to think about whom to identify as the *COMMUNICATION BROKER*. Furthermore, it can be possible to identify multiple *COMMUNICATION BROKERS* within the co-located teams that are responsible for the communication that covers the *COMMUNICATION BROKER*'s working activities, tasks, or professional skills. Hence, an AWARENESS NETWORK of a co-located team can offer valuable clues to identify *COMMUNICATION BROKER*s.

DSD teams utilize various repositories, such as communication, source code, documentation, and bug tracking. Communication repositories, such as mailing lists or online forums, support DSD teams in order to define, clarify and modify the software's requirements. Source code repositories, such as concurrent version control (CVS) or Subversion (SVN), are utilized to version the software's source code. Documentation repositories keep track about a project's relevant documents, such as architecture documents, written deliverables, or change requests. Bug tracking repositories, such as Bugzilla, give information about identified and detected software bugs. According [HS08], mining just one repository is not sufficient to enhance the communication of DSD teams.

6. Future Research Challenges

Nowadays, model-driven development (MDD) is a popular paradigm in software development. Models can be used to define the software's requirements and technical artefacts in a platform-independent way, making it possible to generate recurring and schematic parts of the software automatically [Sch06]. Model repositories, such as EMFStore [KH10] or MORSE [HZD09], store and version the software's models. To the best of our knowledge, there exist no approaches yet that mine model repositories to support the DSD team communication. In our opinion, potential future research challenges exist to (1) mine model repositories and (2) to combine the mining results with the mining results of, for example, communication, source code, documentation, or bug tracking repositories.

A further identified research challenge deals with the broad variety of the repositories' data schemes. For example, the data schema of communication repositories differs from the data scheme of a bug repository. Many existing data mining approaches utilize the repositories' meta-data to construct, for

example, social networks. But, just mining the meta-data is not sufficient to support the DSD team communication. Furthermore, the implementations of the various repository tools differ. For example, the CVS source code repository stores the data differently than the SVN source code repository. We ask if it is possible to define or standardize one universal meta-data and data scheme to ease the mining to improve the DSD teams' communication?

During physical DSD team meetings, the attendees formulate meeting minutes. Mostly, these do not cover all discussions during the meetings and are difficult to propagate to the team members who did not attend the meeting. Furthermore, personal communication among co-located team members, such as during coffee breaks, at the hallway, or just in the office, are difficult to propagate to the other co-located team members and especially to the remote team members. An important question is how to catch such conversations and how to propagate the information to the appropriate co-located and remote team members?

7. Conclusion

Communication in distributed software development (DSD) teams is challenging because several questions arise that usually do not arise in co-located software development teams. Successful communication within DSD teams implies bigger success of the developed software. In this paper, we present three patterns to enhance the DSD team communication. We mined the patterns from surveying state of the art approaches published at several conferences and books that focus on DSD.

The *COMMUNICATION BROKER* pattern can improve the communication between co-located and remote DSD teams. The *AWARENESS NETWORK* pattern helps to find appropriate DSD team members with some desired professional skills. The patterns can help to organize DSD teams better in order to develop software more successfully.

Acknowledgements
We want to thank Lise Hvatum. Lise gave us great hints and suggestions to define and to improve the patterns and the paper.

References:

[Ale77] Christopher Alexander, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, 1977.

[CM03] D. Cubranic and G. C. Murphy. **Hipikat: Recommending Pertinent Software Development Artifacts.** In Proceedings of the *25th International Conference on Software Engineering (ICSE '03)*, pages 408–418, Washington, DC, USA, 2003. IEEE Computer Society.

[Con68] M. Conway, **How Do Committees Invent?**, Datamation, 14(4):28–31, April 1968.

[CWH+06] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. **Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools.** In Proceedings of the 20th Anniversary *Conference on Computer Supported Cooperative Work (CSCW '06)*, pages 353–362, New York, NY, USA, 2006. ACM.

[DB92] P. Dourish and V. Bellotti. **Awareness and Coordination in Shared Workspaces**. In Proceedings of the 1992 *ACM conference on Computer-supported cooperative work* (CSCW '92), pages 107–114, New York, NY, USA, 1992. ACM.

[DIS+07] D. Damian, L. Izquierdo, J. Singer, and I. Kwan. **Awareness in the Wild: Why Communication Breakdowns Occur**. In Proceedings of *The International Conference on Global Software Engineering* (*ICGSE '07)*, pages 81–90, Washington, DC, USA, 2007. IEEE Computer Society.

[DKM10] D. Damian, I. Kwan, and S. Marczak. **Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People**. In *Collaborative Software Engineering*, pages 57–76. Springer Berlin, Heidelberg, 2010.

[DT87] T. DeMarco and L. Timothy. **Peopleware - productive projects and teams**. Dorset House Publishing, New York, 1987.

[EC06] K. Ehrlich and K. Chang**. Leveraging expertise in global software teams: Going outside boundaries**. In Proceedings of the *IEEE International Conference on Global Software Engineering (ICGSE'06)*, pages 149–158, Washington, DC, USA, 2006.

[Eck10] J. Eckstein, **Agile Software Development with Distributed Teams: Staying Agile in a Global World,** Dorset House, New York, 2010 250 pages ISBN 978-0-932633-71-2

[GHJ+95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley Professional, 1995.

[Har99] N. Harrison, **The Language of Shepherding,** http://hillside.net/index.php/the-language-of-shepherding

[Has08] A. E. Hassan. **The Road Ahead for Mining Software Repositories**. In Proceedings of *Frontiers of Software Maintenance (FoSM '08),* pages 48–57, 2008.

[HK05] J. Han and M. Kamber. **Data Mining: Concepts and Techniques**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[HM06] P. Hinds and C. McGrath. **Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams.** In Proceedings of the *20th Anniversary Conference on Computer Supported Cooperative Work* (CSCW '06), pages 343–352, New York, NY, USA, 2006. ACM.

[HRG+08] T. Hildenbrand, F. Rothlauf, M. Geisser, A. Heinzl, T. Kude, **Approaches to Collaborative Software Development.** In Proceedings of the *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '08)*. pages 523-528, IEEE Computer Society, Washington, DC, USA,

[HWC06] L. Hossain, A. Wu, and K. K. S. Chung. **Actor Centrality Correlates to Project Based Coordination**. In Proceedings of the 20th Anniversary *Conference on Computer Supported Cooperative Work (CSCW '06)*, pages 363–372, New York, NY, USA, 2006. ACM.

[HZD09] T. Holmes, U. Zdun, and S. Dustdar. **MORSE: A Model-Aware Service Environment**. In Proceedings of the 4th *IEEE Asia-Pacific Services Computing Conference (APSCC '09)*, pages 470–477. Dec. 2009.

[KH10] M. Koegel and J. Helming. **EMFStore: A Model Repository for EMF Models**, In Proceedings of the 32nd *ACM/IEEE International Conference on Software Engineering - Volume 2, (ICSE '10)*, pages 307–308, New York, NY, USA, 2010. ACM.

[Lan09] F. Lanubile. **Collaboration in Distributed Software Development**. In *Software Engineering*, A. Lucia and F. Ferrucci (Eds.). Lecture Notes In Computer Science, Vol. 5413, pages 174-193, Springer-Verlag, Berlin, Heidelberg, 2009.

[MH02] A. Mockus and J. D. Herbsleb. **Expertise Browser: A Quantitative Approach to Identifying Expertise**. In Proceedings of the *24th International Conference on Software Engineering (ICSE '02),* pages 503–512, New York, NY, USA, 2002. ACM.

[MHG+10] I. Mistrik, A. van der Hoek, J. Grundy, and J. Whitehead (Eds.), **Collaborative Software Engineering**. Springer, 2010.

[RJ02] L. Rising, N. S. Janoff. **The Scrum Software Development Process for Small Teams**, in *IEEE Software*, Vol. 17, No. 4. August 2002), pp. 26-32.

[Sch06] D. C. Schmidt. **Model-Driven Engineering**. *IEEE Computer*, 39(2), February 2006.

[SIG+05] A. Scott, L. Izquierdo, S. Gupta, R. Elves, and D. Damian. **Leveraging design patterns in global software development**. Proc. of the Int. Workshop on Distributed Software Engineering, Paris, Aug. 2005.

[SR07] C. de Souza and D. Redmiles. **The Awareness Network: To Whom Should I Display My Actions? And, Whose Actions Should I Monitor?** In Proceedings of the *10th European Conference on Computer-Supported Cooperative Work (ECSCW '07)*, pages 99–117, Springer London, 2007.

[WF94] S. Wasserman and K. Faust. **Social Network Analysis: Methods and Applications**. Cambridge University Press, 1994.

[WSD+09] T. Wolf, A. Schröter, D. Damian, L. D. Panjer, and T. H. D. Nguyen. **Mining Task-Based Social Networks to Explore Collaboration in Software Teams**. *IEEE Software*, 26:58–66, January 2009.