Supporting Distributed Software Engineering in a Fully Distributed Organization

Kevin Dullemond
Delft University of Technology
IHomer
The Netherlands
k.dullemond@tudelft.nl

Ben van Gameren
Delft University of Technology
IHomer
The Netherlands
b.j.a.vangameren@tudelft.nl

Rini van Solingen
Delft University of Technology
The Netherlands
d.m.vansolingen@tudelft.nl

***Abstract*—Software engineering is increasingly carried out in distributed settings. Software engineers are becoming more nomadic in carrying out their work, working from the customer location, the headquarters of their own company, their home, or sometimes even from their holiday locations. Technological support is needed to overcome the negative impacts of distance that are introduced by this trend. The central theme in this paper for supporting dislocated software engineers lies in increasing their awareness level to a level similar to (or even exceeding) what they experience in a co-located setting. In this paper we present the context in which we are bootstrapping a custom fit environment to support a team of fully dislocated software engineers and the incremental process we use. By working in this fashion we are discovering the requirements to support fully distributed teams while at the same time providing our setting with working solutions to help them with their day to day challenges. Finally, this continuous practical use also provides us with empirical data to validate the increase in awareness levels of dislocated software engineers and helps us in pinpointing important open research challenges.**

***Keywords*-Computer-supported collaborative work; Distributed software development; Awareness; Tools and environments; Agile software development**

I. Introduction

In this paper we present our approach to developing and validating technological support for distributed software engineers. The main objective of this paper is not so much on presenting the solutions themselves, but mostly on explaining the setting in which we are bootstrapping a solution that fits a fully distributed setting, and the process we use for doing so. The angle we take in this process focuses on aiding distributed software engineers to acquire a sufficient level of awareness independent on whether they are co-located or dislocated. With awareness we mean *'an understanding of the activities of others, which provides a context for your own activities'* [1]. Having sufficient awareness is essential because software engineering is an inherently cooperative activity which requires potentially many developers to coordinate their efforts to produce a system. In order to do this there exists a need for a shared understanding both about the project itself, like its state and its artifacts, and about the people who work on the project, like their activities, availability and interactions. Acquiring and sustaining this shared understanding is far harder in a distributed setting than in a co-located setting where it is shared relatively passively and unobtrusively [2], [3]. This is why we focus on facilitating this relatively passively and unobtrusively in a distributed setting as well.

We approach this problem by creating a cross-platform, web-based, extensible communication framework which we co-develop with a Dutch software development company and roll out in this setting as well. The company has the policy for its developers to work from home as much as possible and therefore is ideal for identifying the problems faced when working distributed from your colleagues as well as verifying the viability of the solutions produced. We build this platform in two week iterations at the start of which we decide what to build based on what the users find most valuable and at the end of which we perform an evaluation. From the first iteration onwards all the developers use the system in their daily work which enables us to acquire feedback quickly. Working in this fashion provides for a potentially short turnaround time for research subjects as identification, implementation and evaluation can be as quick as a single iteration.

The rest of this paper is structured as follows. In section II we discuss the growing popularity of distributed teams and how collaborating in such teams is more difficult than collaborating co-located. Next, in section III we discuss which awareness needs are reasonably well supported by existing solutions and which are not. Following this, in section IV, we elaborate on the approach we take in this research by discussing (A) the objectives and the reasoning behind these, (B) the setting, (C) the process we use and (D) the implementation of the system. Finally we present a summary and discuss future work in section V.

II. Awareness in Distributed Software Engineering Teams

Collaborative software engineering is increasingly conducted outside of the traditional single office building for instance in multiple dislocated office buildings or from home. This is the result of the increasing globalization of business [4], [5], [6] and the rising popularity of working from home [7]. Advantages of the globalization of business include: market-proximity [8], [9], reducing time-to-market

by working around the clock [4], [10], flexibility with respect to business opportunities [4], [11], reducing costs by delegating work to countries with low labor cost [12], [9] and being able to fully utilize available resources [5], [9]. Advantages of working from home include: increased autonomy [13], increased flexibility [13], increased productivity [14], increased motivation [15] and improvement in the quality of the environment [13].

In collaborative software engineering, having access to the knowledge about the context in which you are working (commonly referred to as *'awareness'* [3], [1]) is essential to properly cooperate with others [3], [16]. Ko et al. [17] reported the most frequently sought information by software developers includes awareness about tasks, artifacts and co-workers. For example, their results show that developers frequently seek information about changes in artifacts they depend on, the activities of their team members and information relevant to their current task. In general there are three strategies to keep people aware of important information during collaboration: polling, alerts and peripheral awareness [18]. Making the information available by polling involves making the information accessible and allowing people to explicitly poll this information on an as needed basis. Using alerts involves intentionally interrupting people to provide awareness information. The main benefit of using alerts is that the recipient can be sure he is notified in time of important information while the main disadvantage is that it can disrupt the recipient from his current task. The final strategy to make awareness information available is peripheral awareness. This involves making information available in the recipient's periphery such that he has access to the information without it distracting him.

In co-located teams all three of these strategies are used. Important examples of methods of sharing awareness in such a setting are meeting in the hallway [19], watching another developers carry out their task [20] and observing changes being made to artifacts [21]. Also, Perry et al. [22] reported that developers spent over half their time interacting with colleagues and that most of the communication is intended to maintain awareness. In distributed teams using such methods to maintain awareness is far more difficult. In their review Omoronyia et al. [23] conclude that: *"overall, the literature suggests that many aspects of awareness that are disseminated as a natural by-product of co-located working are difficult to achieve in distributed working environments."* Therefore it is important that technological support is developed that supports this process so that the advantages of distributed development can be fully exploited. Omoronyia et al. also claim that: *"to leverage the advantages of both co-located and distributed development, it is important that tool support for distributed teams aims to emulate the attributes of co-location awareness."*

III. Awareness and Technological Support

In this section we will closely follow Omoronyia et al. [23] because while there exist other reviews (e.g. [24], [25]) this is the only review of existing technological support for awareness in distributed software engineering teams which links awareness types and their support requirements and in turn cross references awareness systems with these support requirements. Omoronyia et al. choose to focus on five types of awareness that are particularly relevant to supporting group dynamics that exist during collaborative work [26], [27] and use these to structure the analysis of existing technological support. The requirement categories are [23]:

1) Workspace Awareness: *The up-to-the-minute knowledge of other participants interactions with the shared workspace* [28]
2) Informal Awareness: *The general sense of who is around, what they are doing, and what they are going to do* [26]
3) Group-Structural Awareness: *Knowledge about peoples roles and responsibilities, their positions on an issue, their status, and group processes* [26]
4) Social Awareness: *Information about the presence and activities of people in a shared environment* [29]
5) Context Awareness: *The evolving internal and external state information that fully characterizes the situation of each entity in a shared environment* [23]

Omoronyia et al. analyze how well current solutions support awareness by comparing a set of these solutions with a list of awareness elements these can support. This list of awareness elements is created by extending a list of awareness elements defined by Gutwin et al. (see Table 1 in [26]) with additional elements based on the specific five awareness types they consider (see Table II in [23] for this list). The selection of tools to consider was arrived at by going over a range of techniques that have been used to enhance awareness during distributed software development within IDEs and related tools. The techniques they considered were: social tagging, mining relationships, monitoring interactions, a combination of mining relationships and monitoring interactions and, finally, including the notion of time with that combination. The resulting summarizing table depicting which awareness elements the different tools that are discussed support is shown in Table III in [23].

The main benefit of this table is that it helps to identify those elements that are reasonably well supported by existing solutions and techniques and the elements that are not. Looking at the table we see that most of the elements are supported by more than one of the considered tools. The main exceptions to this are the location of developers, the extent to which they are available and all but one of the elements related to context awareness. Therefore these seem like good areas to direct further research.

IV. Approach

In this section we discuss our approach to researching supporting awareness with technology. First we discuss the main objectives we want to achieve and the reasoning behind these objectives. Following this we discuss how we aim to reach these objectives by describing the setting in which the solution will be evaluated and the process we employ in developing and evaluating the solution in that setting. Finally we also briefly discuss how we are implementing the proposed system.

A. Objectives

The goal of our research is to determine how best to support distributed software engineering with technological support for aiding people to acquire sufficient awareness. The two core objectives of our approach to achieve this are the following:

1) Support all awareness needs of software developers in a single platform
2) Enable integration of the awareness information from different information sources

We arrived at these objectives as follows. Having researched the extensive literature of existing attempts at supporting awareness for distributed software engineers (and having developed and evaluated our own solution for being aware of conversations of colleagues, see [30], [31], [32]) we identified a pitfall a lot of these approaches suffer from. Most tools are designed to help resolve a specific type of question and target a specific software artifact. Sillito [33] reports on an empirical study on how programmers resolve change tasks and how tools support them in answering questions they have in the process of carrying out these tasks. He reported that most of the tools that he researched treat questions as if they are asked in isolation while they often are, in fact, part of a larger process. Examples are asking questions on different levels of abstraction and asking questions involving different information sources. Because awareness questions are often part of a larger process, involving a series of questions and activities that provide context, it is often difficult for distributed software engineers to obtain sufficient contextual awareness [23]. Therefore we feel it is highly valuable for a tool supporting awareness for distributed software engineers to facilitate combining the different types of awareness information. Firstly, we propose to do this by providing a single platform to support all awareness needs of distributed software engineers. Secondly, Sillito [33] also states that even programs that do support asking different questions generally fail at combining the information in a useful way and merely report the information in isolation as largely undifferentiated and unconnected lists. Therefore we also think it is important to enable the integration of the awareness information from different information sources. An example of a good approach in enabling such integrated support for a broad range of awareness elements are extensible plugin architecture as can be found in the Jazz and Eclipse IDEs.

B. Setting

The development and evaluation of the solution we are creating is relatively unique because it is done at a company called IHomer, a Dutch software engineering company founded in August of 2008, which is distributed in the true sense of the word. The company works with highly responsible, proactive people, which is also reflected in how it calls the people that form it: participants (instead of employees). It is also reflected in responsibilities because all participants are responsible for all business decisions like the strategy, vision and core values, in contrast with employees at 'regular' companies who are mainly responsible for the specific role they fulfill. As of the time of this writing the company employs 18 participants.

In the company, physically distributed collaboration is common since participants aim to work from home as much as possible. This makes the company a particularly suitable setting for performing this research because of two reasons. Firstly, the people are quite experienced with dealing with the difficulties of working distributed from each other and therefore have quite a good understanding of what is needed to improve this situation. Because of this we can closely collaborate with the other participants to determine which types of awareness are most beneficial to support first and what are good ways to achieve this. Next to this, the other participants also collaborate with us in realizing the actual technical implementations which improves the quality of the solutions and reduces the time it takes to realize these. Secondly, we can also perform high quality evaluations because the company perfectly matches the target setting for which we are attempting to solve issues: a fully distributed organization. These evaluations can also be done in a lightweight manner and with low turnaround time because of the high quality feedback the other participants can give us due to their experience with distributed collaboration. Finally, because the participants encounter the issues we are attempting to solve in their daily work the solutions will also benefit them directly.

C. Process

Based on the specific characteristics of the project, the setting we are conducting research in and our own experiences in the past, the process we use should fulfill three requirements. Firstly, it should be able to cope with uncertainty and changing requirements since we are creating a genuinely novel product and projects creating genuinely novel products are often faced with uncertainty regarding both requirements and implementation technologies. Secondly, it should involve the intended users of the system as strongly as possible because they are quite experienced

with working in a distributed setting since this is something they encounter on a daily basis. Finally, it should stimulate the usage of the platform by all users by providing value as soon as possible. This is important because in earlier research [32] we found that the value of awareness sharing technology (CSCW groupware) is higher when a larger portion of the team uses it and that this is often a problem when introducing such tools in practical settings.

We elect to use an agile process methodology to realize the platform we are creating because such a methodology fulfills all three of these requirements. Firstly, such a methodology is better able to cope with uncertainty and changing requirements in projects than plan-driven approaches [34]. The main ways in which this is accomplished is by acquiring rapid feedback from the actual users of the system by using short iterations, rapid deployment and working closely with the customers. Working closely with the customers is done to acquire feedback but also to discover how value can be created as quickly as possible resulting in increased customer satisfaction and commitment. The specific methodology we elect to use is Scrum [35], [36] an agile process which emphasizes a set of project management values and practices [37]. It does not define any specific software development techniques for the implementation phase but concentrates on how the team members should function in order to produce the system in a flexible way in a constantly changing environment [38].

In the specific way we have implemented the Scrum process in our case, we work with two-week iterations, referred to as sprints in Scrum. An overview of how our sprints looks like is depicted in Fig. 1. Each sprint starts with a sprint planning meeting in which we decide what to do in the sprint based on the product backlog, which is a prioritized list of features for the product, and an estimate of the amount of work of the different user stories on the product backlog: we decide on the sprint backlog. At the end of each sprint we perform a sprint review and a sprint retrospective. The goal of the sprint review is to discuss what has been done in the sprint and compare this to what was agreed upon in the sprint planning meeting at the start of the sprint. The sprint review revolves around reviewing the product and also includes a demonstration of the product. The sprint retrospective revolves around reviewing the process and is intended improve this in the next sprint.

For practical reasons we hold the sprint review and sprint retrospective of one sprint on the same day as the sprint planning meeting of the next sprint. We do this because we require the same group of people to be present at all three meetings, namely the product owner, the development team and a specific unchanging subgroup of the stakeholders. The stakeholders are all the participants at IHomer because these are all users of the system we are creating. The product owner is one of them, and he is responsible for representing all stakeholders and making decisions based on this responsibility. We have decided to include a subgroup of other stakeholders, next to the product owner, in the sprint review and sprint planning meeting to make it easier for the product owner to determine the point of view of the other stakeholders and make decisions based on this. Further, we have included the subgroup of stakeholders in the sprint retrospective as well because in the way we implemented the process, with continual deployment and direct feedback from the stakeholders, the stakeholders are an important and direct part of the process and aspects of the process which include them should be analyzed and improved upon as well.

During a sprint we perform a daily Scrum and release a new built every day. The daily Scrum is a 15-minute time-boxed event for the development team and the product owner to synchronize activities and create a plan for the next 24 hours. This is done by inspecting the work since the last daily scrum and forecasting the work that could be done before the next one. We release a new built every day to provide value and acquire feedback as soon as possible. Two times during a sprint we perform backlog grooming, once on the half-way point and once at the end, on the day before the sprint review, retrospective and planning meeting. Backlog grooming is the act of adding detail, estimates, and order to items on the product backlog. The final role in our process is that of the Scrum master, who is responsible for ensuring the process is followed and impediments are removed. This role is performed by the different members of the development team on a rotation basis.

To provide all stakeholders of the project real time insight in the project and the progress we work with a Scrum-board which we maintain and share using Trello[1], a collaboration tool which organizes projects into boards. On such a board items move along various stages of progress. We use the following stages:

- *Idea Box*: All stakeholders can insert ideas here, but also problems they encounter like actual bugs or other types of feedback. Items can be picked up and put on the product backlog when appropriate
- *Product Backlog*: Prioritized list of features for the product
- *Sprint backlog*: list of work items that are (going to be) implemented in the current sprint
- *Under development*: Work items that are currently being worked on
- *Deployed in current sprint*: Work items completed in the current sprint that are already deployed in the live system

Items can be inserted on various stages on the board based on their actual current status. We differentiate between six types of items on this board:

[1]www.trello.com

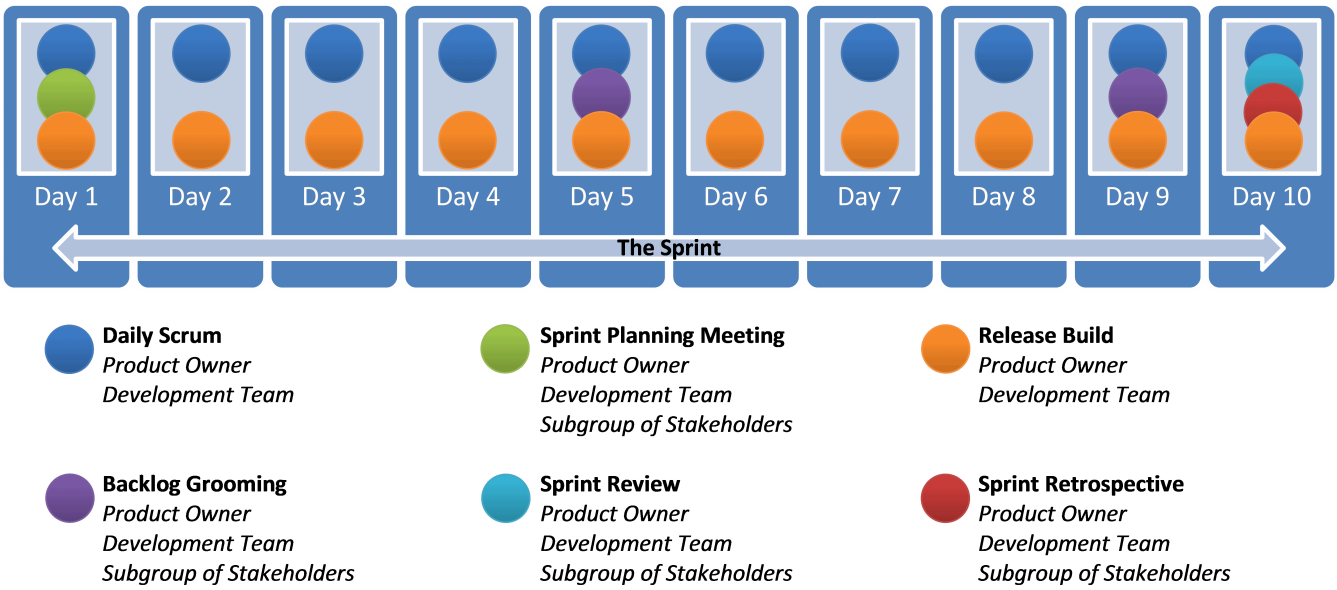

Figure 1: Sprint Overview

- *Feedback*: Reaction to how the system currently functions and explanation of why this is insufficient/sub-optimal
- *Idea*: A rough idea of an addition or alteration to the product backlog
- *User Story*: One or more sentences in the everyday or business language of the end user that captures something the user wants to achieve using the system
- *Defect*: Report of behavior of the system that is in contrast with how the system should function
- *Task*: A unit of work generally between four and sixteen hours which contributes to a backlog item
- *Research Task*: Because we also perform research and write papers in this project we place these on the backlog as well to increase transparency and facilitate sprint planning.

D. Implementation

In section A we discussed the two core objectives of our approach: (i) support all awareness needs of software developers in a single platform and (ii) enable integration of the awareness information from different information sources. These objectives place some constraints on our choice of implementation technology. Firstly, the technology should be cross platform because we want to support all awareness needs in a single platform and our stakeholders use a variety of systems such as Linux, Windows and Mac OS, but also a number of Mobile Operating Systems running on tablets and phones. Therefore the three main options we considered were HTML 5, Adobe Flex and Java FX 2. We choose to go with Adobe Flex because the communication functionalities of HTML 5 are not yet supported in popular web browsers and because Java FX 2 is not yet available for Linux and Mac OS as a stable release. Adobe Flex on the other hand works on all major desktop and mobile platforms, either as a browser plugin or as a standalone Adobe AIR application. The second constraint originating from the single platform objective is that the technology should allow for scalability. We choose to support scalability by using peer-to-peer communication between the users of the system as much as possible. In combination with our choice for Adobe Flex we have chosen to use Real Time Media Flow Protocol (RTMFP) to facilitate the peer-to-peer communication. Our second objective leads to a third constraint on the technology we choose, namely that this should be extensible to facilitate the integration of awareness information from different information sources. To achieve this we have chosen to use a Model-View-Controller framework with an event bus: Mate[2]. Using the Model-View-Controller design pattern results in more extensible code because the model, view and controller are split into separate, loosely coupled components which makes it possible to adapt the flow of the application without changing the model or the view. Using an approach with a central event bus also leads to a more extensible solution since this makes it possible to extend an application without altering the existing solution.

So far, we have performed two sprints and our stakeholders are using a version of the system in which we support the following: (i) standard synchronous communication methods: text, audio and audio combined with video, (ii) showing the availability of the other users (available/unavailable),

[2]http://mate.asfusion.com

(iii) showing the current activity of the other users and (iv) showing the location where the other users are going to work tomorrow. We have chosen to start by supporting standard synchronous communication because our stakeholders identified this as highly valuable and because software engineers spend a large portion of their time communicating [39], [22]. The other three functionalities arise directly from needs our stakeholders expressed. They want to know who is available to decide whether or not to try and contact someone, they want to know someone's current activity to be able anticipate on this and they want to know where colleagues work the next day to be able to more easily identify possibilities for working co-located.

V. Summary and Future Work

In this paper we have presented the research we are currently performing to determine how best to support distributed software engineering with technological support by aiding people to acquire sufficient awareness. First we discussed this is an important topic because of the increasing popularity of distributed development and the challenges this causes. Following this we talked about existing work in facilitating distributed development by supporting the sharing of awareness with technological solutions and about what these solutions support well and where there exist possibilities for improvement. Subsequently we talked about our own approach. We did this by describing and explaining our objectives, describing the setting in which we perform our research, discussing the process we use and finally, by talking about the implementation.

The most valuable contribution of this paper is: *The description of our own unique approach to research how to support awareness in distributed software engineering in which we collaborate with software engineers in a fully distributed company to identify the encountered problems, produce solutions using an agile process and to evaluate these solutions.*

During the first two sprints we have generated some interesting ideas for future work. It would for example be interesting to provide insight into the occurrence and content of the communication of your colleagues. This is both true when you are working together on a project at the same time (e.g. overhearing a conversation of colleagues) but also when you have been away and want to catch up on what happened on the project during your absence. Another example is to let the platform help the users to select the most appropriate form to communicate with their colleagues. Users could for example configure which means of communication they favor in certain situations (e.g. when in a car) and it can also be shown when certain means of communication are currently infeasible (e.g. audio call during a meeting). Similarly the system could also aid in determining who to contact based on a specific question or problem (e.g. who could help me with this specific class?). Finally, it is also possible to show something such as the mood of people in the platform. When working in the same room it is often clear what someone his mood is and when for example someone is unhappy for a prolonged period of time, steps can be taken to find the cause and solve this. When working distributed from each other, such things can go unnoticed and lead to larger problems which can be harder to solve. We already made some first steps in this and users reported they like to share what they are doing and how they feel about this with their colleagues.

References

[1] P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces," in *Proceedings of the ACM 1992 Conference on Computer Supported Cooperative Work*. ACM Press, 1992, pp. 107–114.

[2] J. Fogarty, S. Hudson, C. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. Lee, and J. Yang, "Predicting human interruptibility with sensors," *ACM Transactions on Computer-Human Interaction*, vol. 12, no. 1, pp. 119–146, 2005.

[3] K. Schmidt, "The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'," *Computer Supported Cooperative Work*, vol. 11, no. 3-4, pp. 285 – 298, 2002.

[4] E. Carmel, *Global software teams: collaborating across borders and time zones*. Upper Saddle River: Prentice Hall PTR, 1999.

[5] J. Herbsleb and D. Moitra, "Guest Editors' Introduction: Global Software Development," *IEEE Software*, vol. 18, no. 2, pp. 16–20, 2001.

[6] J. Herbsleb, "Global Software Engineering: The Future of Socio-technical Coordination," in *Proceedings of the IEEE 2007 Workshop on the Future of Software Engineering*. IEEE Computer Society Press, 2007, pp. 188–198.

[7] The Dieringer Research Group Inc., "Telework Trendlines 2009: A Survey Brief by WorldatWork," 2009.

[8] R. Grinter, J. Herbsleb, and D. Perry, "The geography of coordination: dealing with distance in R&D work," in *Proceedings of the ACM SIGGROUP 1999 International Conference on Supporting Group Work*. ACM Press, 1999, pp. 306–315.

[9] D. Damian and D. Moitra, "Guest Editors' Introduction: Global Software Development: How Far Have We Come?" *IEEE Software*, vol. 23, no. 5, pp. 17–19, 2006.

[10] C. Ebert and P. De Neve, "Surviving global software development," *IEEE Software*, vol. 18, no. 2, pp. 62–69, 2001.

[11] J. Herbsleb and R. Grinter, "Architectures, coordination, and distance: Conway's law and beyond," *IEEE Software*, vol. 16, no. 5, pp. 63–70, 1999.

[12] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *IEEE Software*, vol. 18, no. 2, pp. 22–29, 2001.

[13] I. Harpaz, "Advantages and disadvantages of telecommuting for the individual, organization and society," *International Journal of Productivity and Performance Management*, vol. 51, no. 2, pp. 74–80, 2002.

[14] B. Hesse and C. Grantham, "Electronically Distributed Work Communities: Implications for Research on Telework," *Internet Research*, vol. 1, no. 1, pp. 4–17, 1991.

[15] J. Pratt, "Myths and Realities of Working at Home: Characteristics of Homebased Business Owners and Telecommuters," National Technical Information Service, Tech. Rep., 1993.

[16] A. Syri, "Tailoring cooperation support through mediators," in *Proceedings of the 1997 European Conference on Computer Supported Cooperative Work*. Kluwer Academic Publishers, 1997, pp. 157–172.

[17] A. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 2007, pp. 344–353.

[18] J. Cadiz, G. Venolia, G. Jancke, and A. Gupta, "Sideshow: Providing peripheral awareness of important information," Microsoft Research, Collaboration, and Multimedia Group, Tech. Rep., 2001.

[19] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.

[20] L. Segal, "Designing team workstations: The choreography of teamwork," *Local applications of the ecological approach to human-machine systems*, vol. 2, 1995.

[21] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-computer interaction*. Prentice hall, 2004.

[22] D. Perry, N. Staudenmayer, and L. Votta, "People, organizations, and process improvement," *Software, IEEE*, vol. 11, no. 4, pp. 36 –45, Jul. 1994.

[23] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood, "A review of awareness in distributed collaborative software engineering," *Software: Practice and Experience*, vol. 40, no. 12, pp. 1107–1133, 2010.

[24] M. Storey, D. Čubranić, and D. German, "On the use of visualization to support awareness of human activities in software development: a survey and a framework," in *Proceedings of the 2005 ACM symposium on Software visualization*. ACM, 2005, pp. 193–202.

[25] A. Sarma, "A survey of collaborative tools in software development," University of California, Irvine, Tech. Rep., 2005.

[26] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation," *People and Computers*, pp. 281–298, 1996.

[27] T. Gross, C. Stary, and A. Totter, "User-centered awareness in computer-supported cooperative work-systems: Structured embedding of findings from social sciences," *International Journal of Human-Computer Interaction*, vol. 18, no. 3, pp. 323–360, 2005.

[28] C. Gutwin, G. Stark, and S. Greenberg, "Support for workspace awareness in educational groupware," in *The 1995 international conference on Computer support for collaborative learning*. Erlbaum Associates Inc., 1995, pp. 147–156.

[29] W. Prinz, "Nessie: an awareness environment for cooperative settings," in *ECSCW99*. Springer, 2002, pp. 391–410.

[30] K. Dullemond, B. van Gameren, and R. van Solingen, "Virtual open conversation spaces: Towards improved awareness in a GSE setting," in *Proceedings of the 2010 International Conference on Global Software Engineering*. IEEE Computer Society Press, 2010, pp. 247–256.

[31] ——, "Overhearing Conversations in Global Software Engineering - Requirements and an Implementation," in *Proceedings of the 2011 International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2011.

[32] Dullemond, K. and van Gameren, B. and van Solingen, R., "An Industrial Evaluation of Technological Support for Overhearing Conversations in Global Software Engineering," Delft University of Technology, Tech. Rep., 2011, http://www.aspic.nl/publications/TechReport001.pdf.

[33] J. Sillito, G. Murphy, and K. De Volder, "Asking and answering questions during a programming change task," *Software Engineering, IEEE Transactions on*, vol. 34, no. 4, pp. 434–451, 2008.

[34] K. Dullemond and B. van Gameren, "Technological support for distributed agile development," Master thesis, Delft University of Technology, 2009.

[35] K. Schwaber and J. Sutherland, "Scrum guide," *Scrum Alliance*, 2011.

[36] K. Schwaber, "Scrum development process," in *Proceedings of the 1995 ACM Conference on Object Oriented Programming Systems, Languages, and Applications*, 1995.

[37] C. Larman, *Agile and iterative development: a manager's guide*. Addison-Wesley Professional, 2004.

[38] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods. Review and analysis*. VTT Publications, 2002.

[39] J. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481–494, 2003.