Collaboration Spaces for Virtual Software Teams

Kevin Dullemond, Ben van Gameren, and Rini van Solingen,
Delft University of Technology

// Software engineering is a field in which distributed development through virtual teams is a fact of life. Thus, environments for supporting virtual software teams should place collaboration at the forefront. A set of eight core requirements for support environments derived from, and validated in, industrial settings address how to provide virtual software teams with a sufficient level of awareness for their work activities. //

THE DAYS of software engineering taking place in a single office building are long gone; in this dynamic field, virtual software teams are a fact of life. However, bringing distance into software engineering has had an enormous impact on teams and the work itself: distance matters.[1,2]

Nearly everyone with experience in virtual software teams has encountered some challenges. If software engineering were a purely technical, one-man job, we could cope with these challenges much more easily. However, this is a strongly collaborative profession. Social activities represent a considerable portion of the average day of software engineers, who mainly perform four sorts of daily activities:[3]

- coding,
- organizing workspaces and processes,
- representing and communicating design decisions and ideas, and
- communicating and negotiating with various stakeholders.

Note that three of these four are collaborative tasks and not at all technical! Dewayne Perry and his colleagues observed that developers spend more than half their time on activities that include some form of interaction with others.[4] Collaboration is downright essential to software teams,[5] regardless of whether those teams are virtual or not, but distance makes collaboration more difficult, magnifying the challenges that virtual software teams encounter.

Unsurprisingly, virtual software teams have started to develop tools to address these challenges. Today, many environments exist to support virtual software teams, most of which include some support for collaboration. However, these tools have different origins, with different initial goals in mind—compare, for example, integrated development environments such as Eclipse and Microsoft Visual Studio to project-hosting sites such as SourceForge and Google Code and centralized build systems such as Apache Continuum and CruiseControl. All of these tools have been extended with collaborative capabilities; Filippo Lanubile and his colleagues discuss and compare the collaborative support of nine such environments.[6] But a closer examination reveals that the

0740-7459/14/$31.00 © 2014 IEEE

so-called "integration" of collaborative functionalities is limited mostly to making the functionality available in each specific platform separately. Providing functionality such as a mailing list, a message board, or instant messaging is one thing, but integrating these features into a process is quite another.

> To prevent workers from becoming overloaded, it's important to filter the data they get.

Because collaboration is so essential to virtual software teams, we claim that their support environments should place collaboration at the forefront to be successful. Companies such as Atlassian are starting to do this, but we think mobile phone support and related features are only initial steps, and much more is to be gained.

Refocusing Support Environments

In the computer-supported collaborative work (CSCW) community, much research is focused on how to effectively support virtual teamwork.[7–9] One of the most striking conclusions of these studies is that several separate tools support virtual software teams, yet little has been done to integrate them. Moreover, most of the surveyed tools are related to code-specific tasks.[10] In other words, existing support environments for virtual software teams focus mainly on individual programming tasks; any collaborative functionalities are available mostly in isolated (special-purpose) environments, such as a code editor that supports a chat function or a separate debugger that shows which colleagues are using that tool at that point in time.

Distance can't be changed through tools, but the perception of it can be greatly reduced.[11] For instance, virtual teams that have a Google Hangout open all day might be physically distributed, but they won't perceive a large distance within the team. In our research (see the sidebar for an overview), we worked on increasing awareness among virtual software teams through tools. As Paul Dourish and Victoria Belotti defined it, "awareness is an understanding of the activities of others, which provides a context for your own activity."[12] Ideally, tools that support virtual teams need to bring engineers to the same (or a higher) degree of awareness as when they're collocated. When this happens, teams can truly collaborate on a global field, perhaps even making the term *virtual* superfluous.

Requirements for Virtual Team Support Environments

On one hand, collaboration should be a first-class citizen, taking a prominent role at the core of the support environment, but on the other, exchanging awareness information relatively passively and unobtrusively is essential for seamless collaboration. What we need is a single platform that both supports all the awareness needs of virtual software teams and integrates awareness information from different information sources.

But how can we make such an environment a reality? Over the past four years, we worked with real-life virtual software teams to identify their needs and validate possible solutions. We condensed our results into a set of eight core requirements that address how to provide virtual software teams with a higher level of awareness, thereby helping them function as a team more effectively.

Requirement 1: Enable Unobtrusive Awareness Information Exchange

In a traditional collocated setting, awareness is achieved without much effort, but in a distributed one, engineers must manually analyze, filter, and combine available information.[13]

For example, when you're working in an office with a software team, it's easy to know who else is present, what they're doing, and to what extent you can interrupt someone and ask them for help. Tools to support virtual software teams should enable a comparable level of awareness, a goal that's feasible because a system can "know" what task you're working on, who previously edited your file, and whether colleagues can be interrupted. Your system could ideally provide you with contextualized information without you actively searching for it or disturbing you. Of course, such a system would require highly intelligent filtering and a basic understanding of what engineers do.

Requirement 2: Make Basic Work-Related Data Available

Obviously, support environments need to make necessary information available, but the biggest challenge

is to ensure that it's available at the appropriate time. Therefore, such systems should be built on a deep knowledge of the software engineering profession.

Our results showed that virtual software team members consider a large and diverse set of information to be important as long as it's directly related to their current project. That said, receiving updates on a project that you're part of but currently not working on is still valuable, and receiving updates on organizational information items is valuable no matter what relation it has to your current project or task.

Requirement 3: Provide Multisource Data Combinations

To collaborate effectively, team members need to combine information from different sources; through automation, support environments can relieve team members from taking on this burden personally. As Grady Booch and Alan Brown state, automation's purpose is to "create a frictionless surface for development by eliminating or automating many of the daily, noncreative activities of the individual and the team and by providing mechanisms that encourage creative, healthy, and high-bandwidth modes of communication among a project's stakeholders."[14]

One example of the above is combining data from repositories, task boards, and defect systems and providing all of it in one system. Other information such as tweets, calendar items, and ongoing chats can be combined as well. Validating such a (prototype) system proved valuable over the five months we worked with one virtual software team—they felt the system positively impacted their team awareness level.

Requirement 4: Filter Irrelevant Information

To prevent workers from becoming overloaded, it's important to filter the data they get. Environments for virtual software teams should recognize automatically what information is relevant to the current activity. They should also contain a functionality that automatically recognizes when individuals can be interrupted to provide this information.

We investigated the concept of how virtual office walls can be used to contextualize information based on someone's current activity and provided empirical evidence that this approach was indeed an effective method for supporting people in performing their tasks. We also investigated the prioritization of different types of information and how it changes based on current activity. We found that some software engineers want to be informed of a wide variety of information, but they're primarily interested in direct updates about completed artifacts (requirements, design, and verification results) or information about the technological solution itself (technical implementation, selected solutions, and component design). We also found that virtual software team members generally don't mind being interrupted, but they explicitly indicated that they prefer not to be disturbed during activities of a highly interactive nature (customer workshops) or that require a high level of concentration (fixing complex problems).

Requirement 5: Represent and Recognize Current Contexts of Team Members

Support environments must be able to recognize a team member's current context so that they can filter and provide only the information that's valuable to that person at a specific time. Automatically recognizing a team member's current task can be done by assuming that he or she is working on the task personally selected from the digital task board. However, when a team member starts to write an email, is that message related to the task at hand or to something else?

Artificial intelligence is likely needed to fully recognize what's happening in such a scenario. Intense integration with other utilities would be necessary as well—for example, mobile phones (with detection to see if a call is task related or not), office chairs (is the engineer present?), repository activity (for which project is the engineer checking in new code?), and so on.

> Distance can't be changed through tools, but the perception of it can be greatly reduced.

Requirement 6: Support the Overhearing of Conversations

We discovered various benefits and challenges to overhearing conversations in virtual software teams—specifically, the information and useful actions in those conversations. One of the most important benefits of overhearing conversations is

OVERVIEW OF UNDERLYING EMPIRICAL STUDIES

This article is based on our work on boosting the advantages of global software engineering by neutralizing certain disadvantages via technological solutions. We performed this research iteratively, meaning we first studied one aspect, including an empirical evaluation, and then directly applied findings and lessons learned. Table A gives an overview of all the separate studies, including their relation to the requirements in the main text.

We began by investigating the effects of enabling the overhearing of conversations in virtual software teams (studies [S1–S4]). Based on this research, we built a tool called Communico and evaluated it in a four-month case study at Exact, a Dutch software product company with four locations: Malaysia, Belgium, the Netherlands, and the US. We used focus groups, semistructured interviews, a questionnaire, and transactional log analysis on a sample of 47 participants involved in a project that took place on three of the four sites.

In a different company (IHomer, a small Dutch software company), we studied the impact of microblogging with mood indicators ([S5]). This research was based on 13 months of data from a system that enabled virtual team members to share short status updates with their colleagues along with an associated mood metric. We performed content analysis of all the status updates and performed five follow-up interviews to gain more insights.

After this, we turned our research to the challenge of supporting collaboration as a whole—examining the general setting of collaboration versus single isolated issues. We published an article claiming that collaboration should be at the core of support environments for virtual software teams (K. Dullemond, B. van Gameren, and R. van Solingen, "Collaboration Should Become a First-Class Citizen in Support Environments for Software Engineers," *Proc. 2012 8th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2012, pp. 398–405).

Subsequently, we performed several studies that were also conducted at IHomer, where the default daily work location for employees was at home; virtual team members met face-to-face once a week. First, we started working toward a single environment that would provide virtual software teams with all the information they would need to collaborate effectively. We investigated which types of information were valuable based on how related the information was and the current activity ([S6]). We used a focus group study, followed with a survey to collect more information. This survey was replicated outside of IHomer for reconfirmation purposes.

After that, we investigated when software engineers needed specific types of information. We first investigated how to restrict the information based on the software engineer's current activity ([S7] and [S8]). Then we performed a quasi-experiment using an environment in which participants performed simple editing tasks. We divided the 12 participants into two separate groups of 6; the participants assigned to the test group used an environment in which the information was contextualized based on their current tasks, while the control group used an environment in which that wasn't the case. We measured differences in speed and quality, confirming the value of the virtual office wall concept. Afterward, we investigated which tasks software engineers were willing to have interrupted for certain types of information ([S9]). We investigated this with a group of 10 experienced virtual software team members from various companies.

Finally, we constructed a single environment with collaboration at its core and automatic filtering that varied depending on teams, projects, and tasks. Specifically, filtering depended on the current task, team, or project. During code reviews (known from the current task on the task board), new check-in information from the project and team wasn't shown—instead, team member mood changes from positive to negative (or vice versa) appeared. Depending on personal preferences, participants could adjust the filtering rules. This environment isn't finished, but we did incorporate the core findings from the above studies and evaluated them in a five-month case study, collecting data in six interviews ([S10]).

Our studies were empirical but in most cases had only a limited sample size or focused on only two industrial contexts. As such, we remain modest regarding the generalizability of our findings. However, to improve the working environments of virtual software teams, researchers must study their real environments. We encourage others to perform similar studies in other companies. Such findings of real-world practice will build an empirical body of knowledge to fulfill the needs of virtual software teams, now and in the future.

TABLE A

Overview of the studies performed in this research.

ID	Topic	Reference	Requirement in the main text
[S1]	Analytical study about the effects of overhearing conversations in virtual software teams	K. Dullemond, B. van Gameren, and R. van Solingen, "Virtual Open Conversation Spaces: Towards Improved Awareness in a GSE Setting," *Proc. 5th IEEE Int'l Conf. Global Software Eng.*, 2010, pp. 247–256.	6
[S2]	Empirical study about the conceptual value of overhearing conversations in virtual software teams; validating the concept of S1	K. Dullemond, B. van Gameren, and R. van Solingen, "An Exploratory Study on Open Conversation Spaces in Global Software Engineering," *Proc. 7th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2011, pp. 307–316.	6
[S3]	Analytical study about the requirements for and an implementation of overhearing conversations in virtual software teams	K. Dullemond, B. van Gameren, and R. van Solingen, "Overhearing Conversations in Global Software Engineering: Requirements and an Implementation," *Proc. 7th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2011, pp. 1–8.	6
[S4]	Empirical study about the value of enabling overhearing of conversations in virtual software teams; validating the requirements of S3	K. Dullemond and B. van Gameren, "An Industrial Evaluation of Technological Support for Overhearing Conversations in Global Software Engineering," *Proc. 7th Int'l Conf. Global Software Eng.*, 2012, pp. 65–74.	6
[S5]	Empirical study on the usage of a microblogging system with mood indicators for virtual software teams	K. Dullemond et al., "Fixing the 'Out of Sight Out of Mind' Problem: One Year of Mood-Based Microblogging in a Distributed Software Team," *Proc. 10th Working Conf. Mining Software Repositories*, 2013, pp. 267–276.	7
[S6]	Empirical study about which types of information are valuable and their relative importance to virtual software teams	K. Dullemond and B. van Gameren, "What Distributed Software Teams Need to Know and When: An Empirical Study," *Proc. 8th Int'l Conf. Global Software Eng.*, 2013, pp. 61–70.	2, 4, 8
[S7]	Analytical study about restricting information based on the current activity in virtual software teams	B. van Gameren, K. Dullemond, and R. van Solingen, "Auto-erecting Virtual Office Walls," *Proc. 8th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2012, pp. 391–397.	1, 3, 4, 5
[S8]	Empirical study to validate restricting information based on the current activity of virtual software teams, as conceptually designed in S7	B. van Gameren, R. van Solingen, and K. Dullemond, "Auto-Erecting Virtual Office Walls: A Controlled Experiment," *Proc. 8th Int'l Conf. Global Software Eng.*, 2013, pp. 206–215.	4, 5
[S9]	Empirical study to identify during which tasks software engineers can be interrupted and with what kinds of information	B. van Gameren and R. van Solingen, "When to Interrupt Global Software Engineers to Provide Them with What Information?," *Proc. 9th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2013, pp. 495–504.	2, 4, 8
[S10]	Empirical study on a single environment for two virtual software teams (during five months of usage) addressing derived requirements from our previous studies	K. Dullemond and R. van Solingen, "Increasing Awareness in Distributed Software Teams: A First Evaluation," *Proc. 9th Int'l Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2013, pp. 325–334.	3, 7

Rule 1. The more related something is to a software engineer's current project, the greater the importance of receiving updates about it. Also, a larger set of information items is important when it's related to the current project.

Rule 2. Software engineers are primarily interested in direct updates about completed artifacts, especially if they're related to the technological solution. Such updates are also valuable for a project that a software engineer is part of but not currently working on.

Rule 3. Software engineers are less interested in direct updates concerning the procedures and support environments used than about the completed artifacts themselves. The importance of such updates is even lower when a software engineer isn't performing a core activity.

Rule 4. Receiving updates on organizational information is valuable no matter how it relates to the current project. Updates on project-related information are considered more important during work-related travel than during free time or holidays.

Rule 5. Software engineers prefer not to be interrupted when they're performing activities of a highly interactive nature or that require a high degree of concentration.

Rule 6. Even if the information is highly relevant and important, software engineers don't want to be immediately informed of anything when they're performing an activity during which they prefer not to be interrupted. When in doubt, do not disturb!

FIGURE 1. Rules for when to interrupt software engineers.

having access to your colleagues' technical knowledge. At the same time, the greatest challenge is that overhearing your colleagues talk can be a distraction, and conversational context can be unclear.

The most important information about a conversation is its topic, so a support environment should help identify or track it, even if it changes midstream. Furthermore, it should be possible to initiate, participate in, discover, watch, and finish conversations, invite others to join them, and make conversations private (not "overhearable").

Requirement 7: Support Mood Sharing

Being able to have an idea of your teammates' mood and overall happiness is important to virtual software teams. We found evidence for this when we investigated the use of a system that supports microblogging with mood indicators. In particular, we found that being able to express a small amount of information together with their associated mood made people feel more connected with one other on a social level.

This evidence was reaffirmed when we evaluated a system that we developed and designed with collaboration at its core. In this later evaluation, knowing the happiness of colleagues was identified as one of the system's two most valuable aspects.

Requirement 8: Provide for Interruption Support

Virtual software team members need a variety of information about the context in which they're working to collaborate effectively with their colleagues. Support environments have the potential to regulate that information based on both its importance and the current degree of "interruptibility" of team members. To strike this balance, support systems must consider the noises and distractions that software engineers face, even when they're collocated.

We identified the specific activities during which software engineers can be interrupted and which types of information are worth that interruption. On one hand, software engineers need to know immediately about completed artifacts and changes to the technological solution itself, but on the other, during highly interactive activities or those requiring a high level of concentration, they shouldn't be interrupted at all. Figure 1 combines our findings into a set of six interruption rules.

Challenges remain in providing virtual software teams with a support environment that enables the same or higher degree of awareness found in a collocated setting. When the support environment reaches that point, distance will no longer be an issue. We expect a major step in this direction when tool suppliers change their focus from a coding orientation to a collaboration one.

We imagine a future in which team members automatically receive the information they need when they need it and without disrupting them in their core activity. When this future is attained, the degree of awareness will be lifted far beyond being collocated.[15] Why would you travel to work when the actual degree of awareness is better at home?

Acknowledgments

We thank all participants in the presented studies, in particular, the distributed software engineers at IHomer and Exact. We also thank David Redmiles for his help in improving an earlier version of this article. Finally, we thank the IEEE Computer

Society's reviewers and editors for their constructive feedback on earlier versions, helping us get our article into this special issue.

References

1. E. Carmel and R. Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development," *IEEE Software*, vol. 18, no. 2, 2001, pp. 22–29.
2. G.M. Olson and J.S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15, no. 2, 2000, pp. 139–178.
3. J. Strübing, "Designing the Working Process—What Programmers Do beside Programming," *User-Centred Requirements for Software Engineering Environments*, Springer Berlin Heidelberg, 1994, pp. 81–90.
4. D.E. Perry, N.A. Staudenmayer, and L.G. Votta, "People, Organizations, and Process Improvement," *IEEE Software*, vol. 11, no. 4, 1994, pp. 36–45.
5. N. Ahmadi et al., "A Survey of Social Software Engineering," *Proc. 23rd IEEE/ACM Int'l Conf. Automated Software Eng.*, 2008, pp. 1–12.
6. F. Lanubile et al., "Collaboration Tools for Global Software Engineering," *IEEE Software*, vol. 27, no. 2, 2010, pp. 52–55.
7. J. Portillo-Rodríguez et al., "Tools Used in Global Software Engineering: A Systematic Mapping Review," *Information and Software Technology*, vol. 54, no. 7, 2012, pp. 663–685.
8. I. Steinmacher, A.P. Chaves, and M.A. Gerosa, "Awareness Support in Global Software Development: A Systematic Review Based on the 3C Collaboration Model," *Collaboration and Technology*, Springer Berlin Heidelberg, 2010, pp. 185–201.
9. M. Jiménez, M. Piattini, and A. Vizcaíno, "Challenges and Improvements in Distributed Software Development: A Systematic Review," *Advances in Software Engineering*, vol. 2009, 2009, article 710971.
10. B. Sengupta, S. Chandra, and V. Sinha, "A Research Agenda for Distributed Software Development," *Proc. 28th Int'l Conf. Software Eng.*, ACM, 2006, pp. 731–740.
11. R. Prikladnicki, "Propinquity in Global Software Engineering: Examining Perceived Distance in Globally Distributed Project Teams," *J. Software: Evolution and Process*, vol. 24, no. 2, 2012, pp. 119–137.
12. P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces," *Proc. 1992 ACM Conf. Computer-Supported Cooperative Work*, 1992, pp. 107–114.
13. C. De Souza and D.F. Redmiles, "The Awareness Network, to Whom Should I Display My Actions? And, Whose Actions Should I Monitor?," *IEEE Trans. Software Eng.*, vol. 37, no. 3, 2011, pp. 325–340.
14. G. Booch and A.W. Brown, "Collaborative Development Environments," *Advances in Computers*, vol. 59, 2003, pp. 1–27.
15. J. Hollan and S. Stornetta, "Beyond Being There," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, 1992, pp. 119–125.

ABOUT THE AUTHORS

KEVIN DULLEMOND was a software engineer and researcher at the Delft University of Technology when this work was performed. Over the past four years, he studied global software engineering teams to understand what they need in terms of technological support to collaborate effectively. This research can be found in his PhD thesis, "Supporting Collaboration in Global Software Engineering"; www.aspic.nl. Dullemond received a PhD in computer science from the Delft University of Technology. Contact him at kevin.dullemond@gmail.com.

BEN VAN GAMEREN is a software engineer at IHomer and researcher at the Delft University of Technology. Over the past four years, his research focused on how to support global software engineers with technological support for passively and unobtrusively acquiring a sufficient level of awareness. This research can be found in "Auto-Erecting Virtual Office Walls: Constructing a Virtual Office for Global Software Engineers"; www.aspic.nl. Van Gameren received a PhD in computer science from the Delft University of Technology. Contact him at benvangameren@gmail.com.

RINI VAN SOLINGEN is a part-time full professor of global software engineering at the Delft University of Technology, where he heads research and education regarding worldwide distributed (virtual) software teams. He's also CTO of Prowareness (www.scrum.nl). Van Solingen is a member of IEEE. Contact him at d.m.vansolingen@tudelft.nl, follow him on Twitter (@solingen), or read his blog at www.rinivansolingen.com.