[JWCR18] Rodi Jolak, Andreas Wortmann, Michel Chaudron, and Bernhard Rumpe:
Does Distance Still Matter? Revisiting Collaborative Distributed Software Design.
In: IEEE Software, 35(6):40-47, 2018.
www.se-rwth.de/publications/

Does Distance Still Matter?

Revisiting Collaborative Distributed Software Design

Rodi Jolak, Chalmers University of Technology and Gothenburg University

Andreas Wortmann, RWTH Aachen University

Michel Chaudron, Chalmers University of Technology and Gothenburg University

Bernhard Rumpe, RWTH Aachen University

// Studying the design activities of colocated and distributed software designers revealed that despite comprehensive technological improvements, distance still matters. To ensure effective distributed software design, designers must consider extra (nontechnical) details. //

COMPANIES ENGAGE IN global software engineering (GSE) to reduce development time and costs. Companies also head toward cross-site distribution of their development work to take advantage of proximity to markets and customers.[1] However, working at a distance might compromise the effectiveness of GSE.

There are two important challenges to making GSE successful. Almost two decades ago, Gary Olson and Judith Olson raised these challenges:[2]

- *technological challenges* raised by the need for efficient, effective remote-collaboration tools and media; and
- *social challenges* raised by differences in local context, culture, language, and trust between collaborators.

They predicted that future technological advances will reduce the effect of the technological challenges. But they also predicted that working at a distance will rarely succeed owing to the inevitable differences raised by the social challenges. However, advances in communication and collaboration technologies raise the question of whether distance still matters.

One of the key activities of software engineering is software design. It comprises discussing requirements, exploring the problem domain, and making design decisions. When globalized, software design could become less effective. Several design activities could be affected, including

- design modeling (representation),
- design reasoning (about problem-domain and solution-domain design aspects), and
- design communication.

Also, lack of awareness (understanding others' activities) and problems with communication media might threaten the success of distributed software design.

Many researchers have explored the impact of distance on collaborative work. James Herbsleb argued that colocation fosters communication

because developers are aware of who is around and who is doing what.[3] In contrast, being unable to share resources and see what is happening at the other sites hinders communication across different locations.

Pernille Bjørn and her colleagues investigated whether distance still matters for distributed collaboration.[4] They found that the social challenges form an obstacle to achieving effective work between remote collaborators.

Demetrios Karis and his colleagues performed studies of remote collaboration at Google.[5] They found that the use of videoconferencing and video portals contributes to the success of remote collaboration by

- providing presence and status information,
- helping to establish mutual trust and common ground, and
- preventing misunderstandings.

However, when it comes to remote design collaboration, Karis and his colleagues highlighted that developers at Google found collaboration over videoconferencing and video portals a pale imitation of face-to-face interaction. Moreover, the developers complained that the video portals at Google lacked a shared drawing tool to facilitate sketching, designing, and brainstorming.

This conforms with what David Budgen stated in his paper "The Cobblers Children": many modeling tools do not serve the purpose of software design and rarely support realistic software design practices.[6] According to Budgen, modeling tools should preserve the flexibility and simplicity of whiteboards and provide proper support for distributed designers at different locations.

COLLABORATIVE DESIGN MULTIPLE-CASE STUDY

In the first case study, three colocated pairs of software designers worked on a software design challenge at a single location (labeled C1, C2, and C3 in the main article).[14] We conducted the second case study, which involved three design sessions (D1, D2, and D3) between distributed pairs of software design practitioners in Aachen, Germany, and Gothenburg, Sweden. We used the same design problem and timing as in the first case study. This allowed us to

- explore the design decisions and process activities of the two studies,
- gather experiences and seek insights, and
- develop suggestions and recommendations that could be of interest to practitioners concerned with distributed collaborative software design.

The designers in our study varied from three to seven years of professional experience. Three designers worked in automotive software development, two worked in networking solutions, and one worked in traffic technologies. In both studies, the designers solved a software design challenge. The challenge was to create a software design of a simulator that should enable its users to investigate the effects of different signal timing on traffic flow. The challenge description is available in *Software Designers in Action: A Human-Centric Look at Design Work*.[14]

Teams of two professional software engineers solved the same challenge locally, which focused on four functional requirements:

- Users can create a visual map of intersected roads of varying length.
- Users can describe the behavior of the traffic lights at each of the intersections, such that combinations of individual signals that would result in crashes are prohibited.
- Users can simulate traffic flows on the map, and the resulting traffic levels are conveyed visually.
- Users can change the traffic density per road.

We informed the designers that

- their design would be evaluated primarily on the basis of its elegance and clarity, and
- they should focus on the interaction that the users will have with the system, including the basic appearance of the program, and on the important design decisions that form the foundation of the implementation.

To create the design, the designers in our case study used a smart whiteboard with the OctoUML (http://rodijolak.com/#octouml) collaborative-design

Continued

COLLABORATIVE DESIGN MULTIPLE-CASE STUDY (Cont.)

software[15] connected to a computer providing videoconferencing between the two sites. OctoUML is open source; it supports mixed informal modeling (free strokes) and formal modeling (UML class diagram shapes) and supports translating free strokes into class diagram shapes on the fly. It provides predefined shapes, drawing selection mechanisms, and undo and redo functionality. With OctoUML, remote designers share a joint canvas upon which they can draw UML diagrams along with informal elements (i.e., text, drawings, etc.). We chose to deploy a simplified version of OctoUML (a shared canvas and sketching tool) on a smart whiteboard that mimics standard whiteboards (see Figure A). Because the designers in the first case study could not use formal modeling, we deactivated those features as well.

Each design session finished with a questionnaire on the experiences and challenges of collaborative distributed design. We analyzed approximately 10 hours of design activity by six pairs of professional software designers and performed a manual coding of more than 2,000 discussion events. For coding the (conversation) actions of the design sessions, we used the collaborative conversation skill taxonomy of Margaret McManus and Robert Aiken[10] and the design-reasoning decisions of Rainer Weinreich and his colleagues,[11] as presented in Figure 4 in the main article. The former captures various collaborative problem-solving conversation discussions; the latter captures decisions from the problem domain (traffic flow) and solution domain (software engineering). We focused on exploring design reasoning, design communication, awareness, and the number and nature of problems that occurred during the distributed software design sessions.

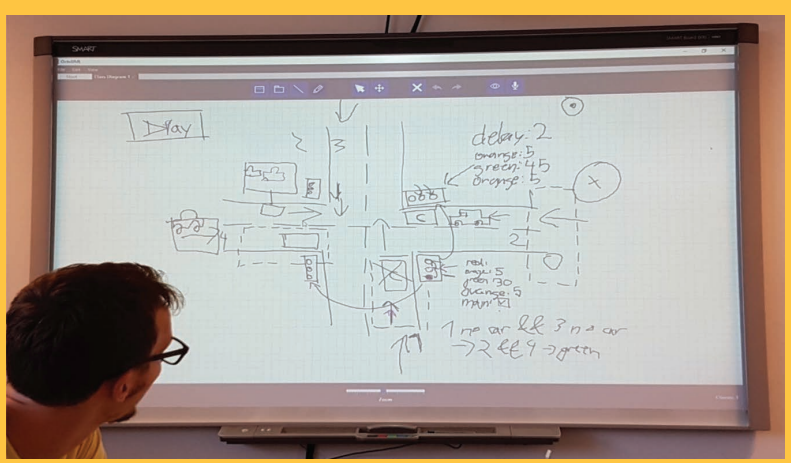

FIGURE A. UI sketches produced by one of the teams.

Several researchers have proposed next-generation design-support tools that are in line with Budgen's guidelines. One of these tools is OctoUML.[7] OctoUML allows mixed informal modeling (sketching) and formal modeling and supports collaborative distributed software design.

To answer the question posed in our article's title, a deep investigation of current practices of collaborative software design is required. To do so, we analyzed a collaborative-design multiple-case study based on two exploratory cases. Details regarding that study are in the sidebar.

How Distance Affects Design Decisions

First, let's look at the type of design decisions that were made and see whether they differed between distributed and colocated design sessions. The graphs in Figure 1 indicate that the colocated designers discussed more design decisions in the problem domain than the distributed designers did. More details on how these design decisions were made are available at http://rodijolak.com/DoesDistanceStillMatter.html.

The decisions in the problem domain consisted mainly of assumptions, as shown in Figure 2. One of the reasons that might have allowed colocated designers to discuss more problem domain design decisions is that they implicitly knew (via facial expressions and body language) whether a specific assumption was mutually understood. In a collaborative process, the conversation can continue only when the collaborators mutually establish what they know.[8] Distance obstructs the process of establishing a mutual understanding of the problem domain between distributed designers. When one designer makes an assumption

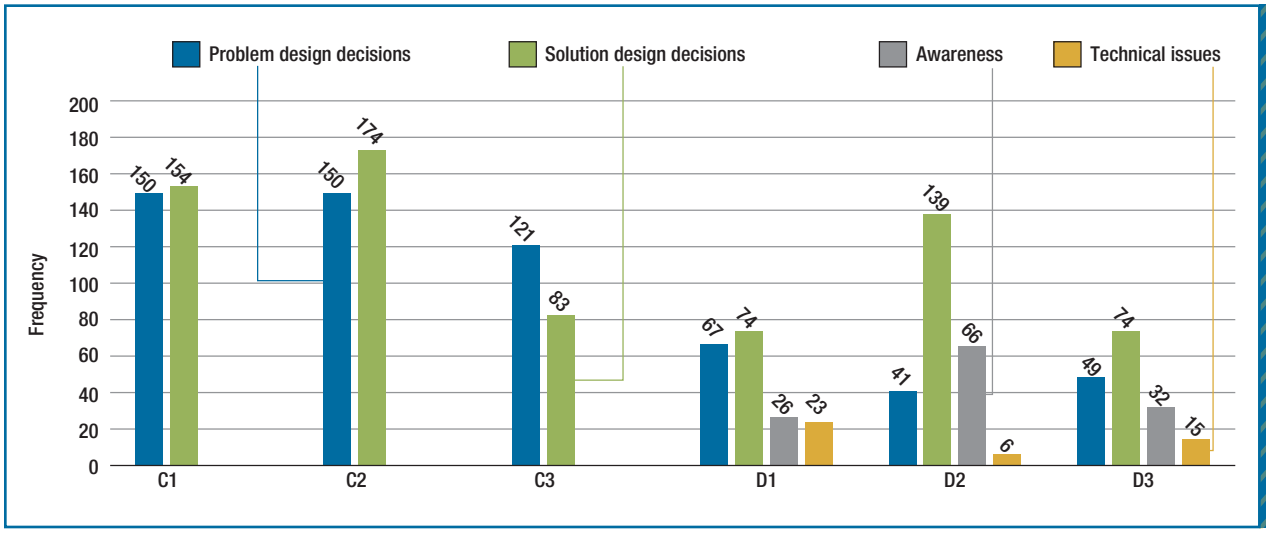

FIGURE 1. The number of design decisions and social and technical issues per each collocated team (C1–C3) and distributed team (D1–D3).

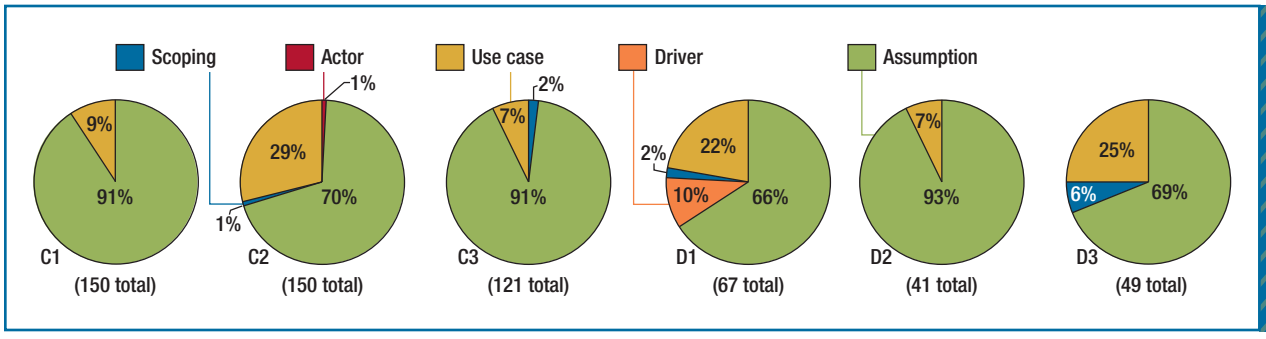

FIGURE 2. The categories of problem domain design decisions made in each design session.

and implicitly perceives that the co-located partner did not understand that assumption, that designer might rephrase the assumption or build more knowledge around it.

In contrast, distributed designers usually do not see each other when discussing assumptions. Hence, the perception of having a mutual understanding (via body language) was rarely possible. Indeed, the distributed designer making an assumption often implicitly considered that the remote partner understood it, thus producing fewer problem domain design decisions.

Technical issues also affected the distributed design discussions—e.g., through blurriness of the voice and instability of the communication medium. Lack of awareness could have also led to fewer problem domain design decisions in the distributed setting. This is because not perceiving another person's actions makes it difficult to initiate contact and often leads to misunderstanding of communication content and motivation.[3]

How Distance Affects Collaborative Communication

The graphs in Figure 3 show how distance could affect communication in distributed design. We see that distributed teams had fewer creative-conflict discussions but more conversation. Creative-conflict discussions can promote software design reasoning and enhance the effectiveness of group tasks.[9]

The creative-conflict problem-solving discussion skill comprises

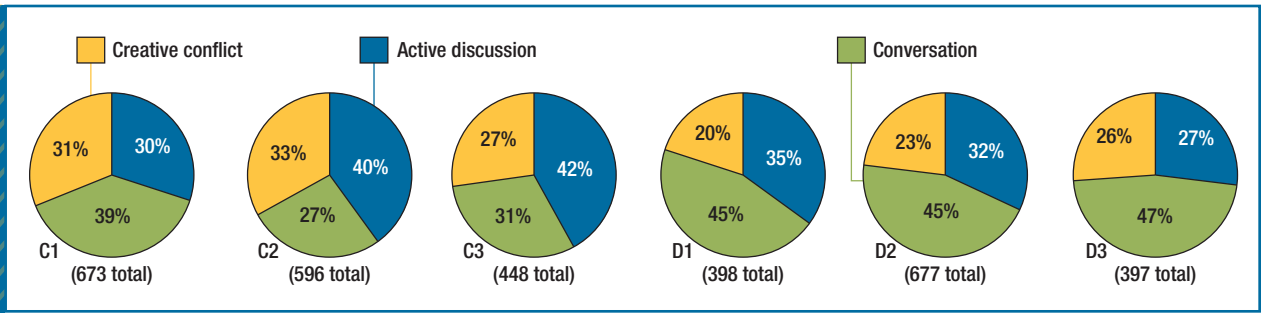

FIGURE 3. The categories of collaborative discussions made in each design session.

two major subskills: Mediate and Argue (see Figure 4). Argue comprises different actions (agree, disagree, offer an alternative, propose an exception, etc.). Distributed designers argued less, as shown in Figure 5. One of the reasons for fewer arguments—and hence fewer creative-conflict discussions—might have been lack of trust. Colocated designers share experiences and context, which helps them to develop trust. Trust is needed for collaborators to be able to challenge each other without frustrating collaboration. Distributed settings can complicate establishing trust and might compromise reliability between the remote collaborators.[12]

Another reason could have been lack of common ground—i.e., the knowledge that the designers are aware of and have in common—in distributed design sessions. When common ground is missing, it might affect distributed collaborators' activities and communication effectiveness.[13] Indeed, this might promote mutual tacit acceptance of design decisions. Hence, it reduces creative-conflict discussions.

Lack of awareness can also reduce creative-conflict discussions. For example, information on authorship (who did what) and intention (what designers are going to do) wasn't available for the participants in our study.

As we mentioned before, more conversation happened between the distributed designers than between the colocated designers. To explain this, we recall that conversation comprises three major subskills: Maintenance, Task, and Acknowledge (see Figure 4). Distant collaboration requires more management overhead and discussion about work coordination. Lack of awareness among distributed collaborators also raises more task discussions and maintenance discussions. For example, the distributed designers summarized design decisions to confirm knowledge of what they had done so far. Summarizing also helped them understand the intention of their partners.

In addition, distributed designers had fewer Inform (see Figure 4) discussions than the colocated designers. This indicates that distributed designers tend to give less information about their decisions, which leads to less active discussion of the essence of and rationale for those decisions.

The Challenges of Distributed Design

The distributed designers reported the following challenges.

Technological Challenges

The designers considered connection instability as a challenge. Network problems and high CPU use in the client machines interrupted several design sessions. These devices were simultaneously running OctoUML, screen- and voice-recording software, and telecommunication software, which overloaded them. Consequently, the designers had to wait until communication was reestablished. This situation can be prevented by avoiding such overloads.

Moreover, the distributed designers complained about the quality of voice communication. This depends on different factors: the quality of the Internet connection, the distance from the microphone, and the volume of the speakers. This problem can be alleviated by adopting advanced communication infrastructures, a high-speed Internet connection, and advanced voice management tools.

Nonetheless, many organizations fail to keep pace with technological advances and therefore fail to manage the aforementioned challenges.

Social Challenges

First, the designers perceived the lack of awareness as a challenge. In particular, they felt that the inability to interpret eye contact, body language, and facial expressions affected their decisions and activities. For instance, one designer said that because he could not see how

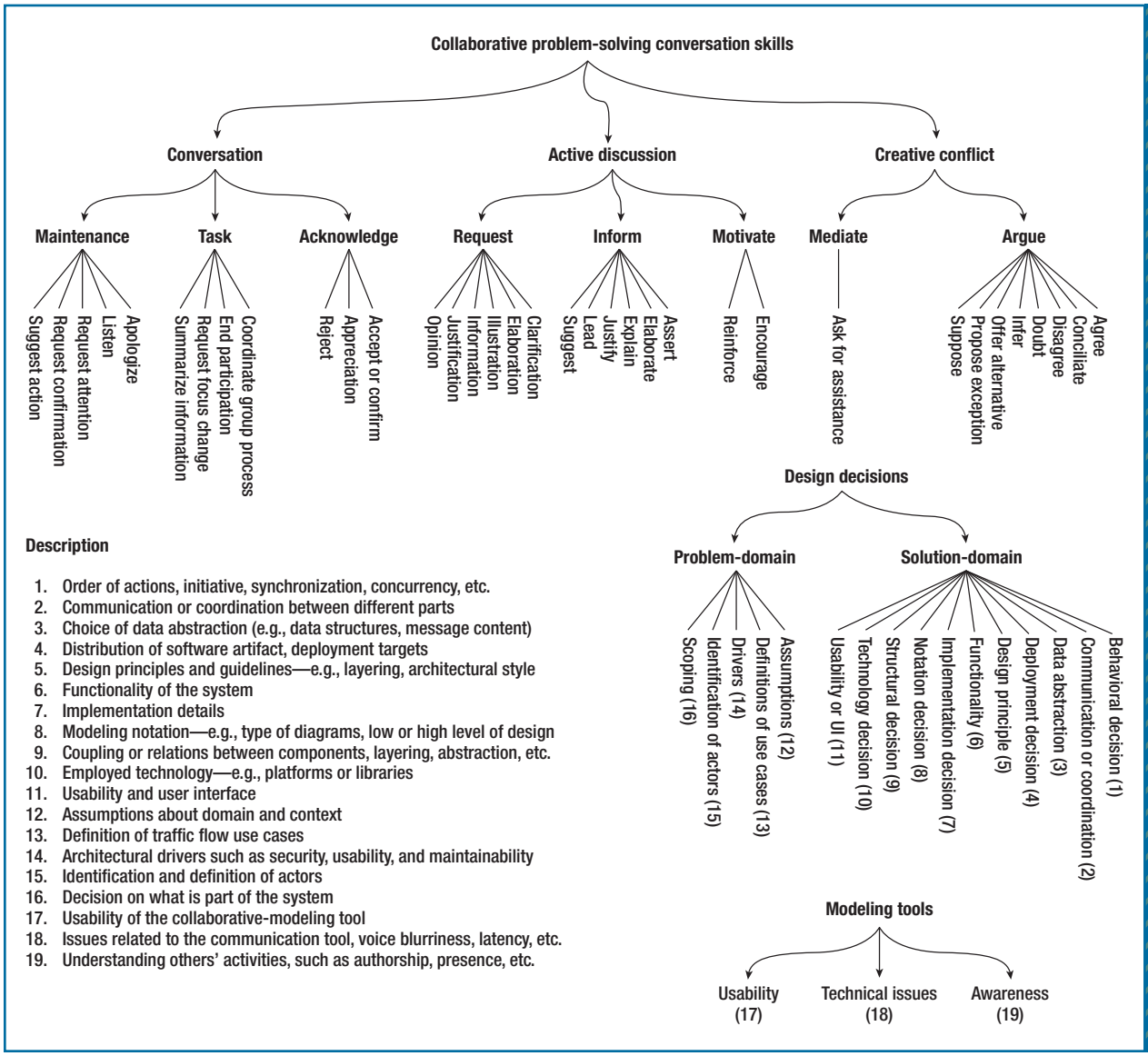

FIGURE 4. A classification schemata for conversation skills and software design decisions.[10,11]

his partner reacted to his proposals, he was unable to decide how to act appropriately. Each designer was unaware of what the collaborating designer was doing and which part of the system that designer was talking about or pointing to.

Second, the designers also perceived the lack of trust as a challenge. In particular, the designers felt that not knowing their collaborator beforehand could have affected their discussions and work.

Other Challenges

The design assignment per se was perceived as a challenge. We believe this confirms our process of thoughtfully planning the assignment to simulate real-world software design situations. This planning took into account ideation, problem domain exploration, and design solution decisions.

The geographical distribution of collaborating partners in practice still raises social and technological challenges. Thus, practitioners should carefully consider whether the distribution is applicable and weigh the benefits of

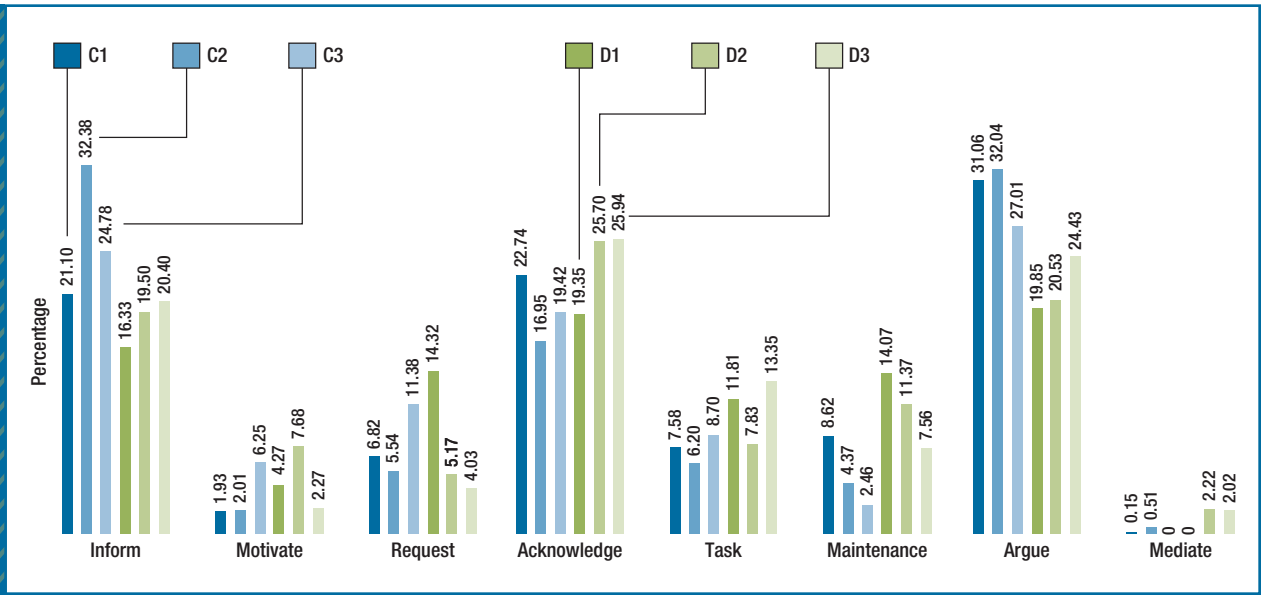

FIGURE 5. Percentages of collaborative-discussion categories per each team.

technology deliberately. To support distributed designing, for instance, modern collaborative-design environments focus on the consistent, real-time sharing of diagrams. However, social awareness, such as the ability of designers to relate to each other through pointing, gaze, and gestures, remains missing.

The designers in our study indicated challenges in distributed collaborations that are beyond the scope of tooling. In particular, in contrast with locally collaborating teams, distributed designers did not know their collaborators beforehand. Hence, they had to build professional and personal trust during the experiment.

On the basis of our results, we encourage software design practitioners aiming to collaborate remotely to consider the following:

- Establish trust via arranging personal or virtual meetings or social events before the remote design sessions.[5]
- Establish common ground via exchanging interests, experiences, expertise, and beliefs between distributed designers.
- Introduce explicit triggers for creative-conflict discussions into the collaboration process.

Furthermore, we recommend that the developers of computer-supported cooperative work (CSCW) tools for software design should support awareness by adapting technology to provide immersive telepresence experiences.

Software design requires extensive exploration of the problem domain and context, and leads up to making critical design decisions about software systems. Moreover, collaborative software design is tightly coupled work that requires either more frequent or more complex interactions.[2] Because of these aspects of software design and because the current technology is still incapable of fully mitigating the social challenges of remote collaboration, we suggest that distance still matters.

References

1. D. Šmite and C. Wohlin, "A Whisper of Evidence in Global Software Engineering," *IEEE Software*, vol. 28, no. 4, 2011, pp. 15–18.
2. G.M. Olson and J.S. Olson, "Distance Matters," *Human–Computer Interaction*, vol. 15, no. 2, 2000, pp. 139–178.
3. J.D. Herbsleb, "Global Software Engineering: The Future of Socio-technical Coordination," *Proc. 2007 Future of Software Eng. Conf.* (FOSE 07), 2007, pp. 188–198.
4. P. Bjørn et al., "Does Distance Still Matter? Revisiting the CSCW Fundamentals on Distributed Collaboration," *ACM Trans. Computer-Human Interaction*, vol. 21, no. 5, 2014, article 27.
5. D. Karis, D. Wildman, and A. Mané, "Improving Remote Collaboration with Video Conferencing and Video Portals," *Human–Computer Interaction*, vol. 31, no. 1, 2016; https://www.tandfonline.com/doi/abs/10.1080/07370024.2014.921506.
6. D. Budgen, "The Cobbler's Children: Why Do Software Design

Environments Not Support Design Practices?," *Software Designers in Action: A Human-Centric Look at Design Work*, M. Petre and A. Van Der Hock, eds., CRC Press, 2013, pp. 199–218.
7. B. Vesin, R. Jolak, and M.R.V. Chaudron, "OctoUML: An Environment for Exploratory and Collaborative Software Design," *Proc. IEEE/ACM 39th Int'l Conf. Software Eng.* (ICSE 17), 2017, pp. 7–10.
8. H.H. Clark and S.E. Brennan, "Grounding in Communication," *Perspectives on Socially Shared Cognition*, L. Resnick et al, eds., Am. Psychological Assoc., 1991, pp. 127–149.
9. A. Soller and A.S. Abu-Issa, "Supporting Social Interaction in an Intelligent Collaborative Learning System," *Int'l J. Artificial Intelligence in Education*, vol. 12, no. 1, 2001; http://iaied.org/pub/980.
10. M.M. McManus and R.M. Aiken, "Monitoring Computer-Based Collaborative Problem Solving," *J. Interactive Learning Research*, vol. 6, no. 4, 1995, p. 307.
11. R. Weinreich, I. Groher, and C. Miesbauer, "An Expert Survey on Kinds, Influence Factors and Documentation of Design Decisions in Practice," *Future Generation Computer Systems*, vol. 47, 2015, pp. 145–160.
12. P.S. Greenberg, R.H. Greenberg, and Y.L. Antonucci, "Creating and Sustaining Trust in Virtual Teams," *Business Horizons*, vol. 50, no. 4, 2007, pp. 325–333.
13. A. Monk, "Common Ground in Electronically Mediated Communication: Clark's Theory of Language Use," *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, J.M. Carroll, ed., Morgan Kaufmann, 2003, pp. 265–290.
14. M. Petre and A. Van Der Hoek, *Software Designers in Action: A Human-Centric Look at Design Work*, CRC Press, 2013.
15. R. Jolak et al., "Towards a New Generation of Software Design Environments: Supporting the Use of Informal and Formal Notations with OctoUML," *Proc. 2nd Int'l Workshop Human Factors in Modeling* (HuFaMo@MoDELS), 2016, pp. 3–10.

ABOUT THE AUTHORS

RODI JOLAK is a PhD candidate in software engineering at the joint Department of Computer Science and Engineering of Chalmers University of Technology and Gothenburg University. His research activities focus on software engineering, software architectures, software design, and human–computer interfaces. Jolak received an MSc in engineering of computing systems from Politecnico di Milano. Contact him at rodi.jolak@cse.gu.se; http://www.rodijolak.com.

ANDREAS WORTMANN is a tenured researcher in RWTH Aachen University's Department for Software Engineering. His research interests include software engineering, software architectures, model-driven development, robotics, and software-language engineering. Wortmann received a PhD in software engineering from RWTH Aachen University. He's a member of IEEE and its Technical Committee on Software Engineering for Robotics and Automation. Contact him at wortmann@se-rwth.de.

MICHEL CHAUDRON is a full professor in the Software Engineering Division of the joint Department of Computer Science and Engineering of Chalmers University of Technology and Gothenburg University. His research interests include software architecture, software design, software modeling, and model-driven software development. Chaudron received a PhD in formal methods and programming calculi for parallel computing from Universiteit Leiden. Contact him at michel.chaudron@cse.gu.se.

BERNHARD RUMPE is the chair of RWTH Aachen University's Department for Software Engineering. His main interests are software development methods and techniques that benefit from rigorous, practical approaches. Rumpe received a habilitation in computer science from the Technical University of Munich. He's editor in chief of *Software and Systems Modeling*. Contact him at rumpe@se-rwth.de; http://www.se-rwth.de/topics.