Design thinking and creativity of colocated versus globally distributed software developers

Received: 1 November 2020 | Revised: 24 May 2021 | Accepted: 1 July 2021

DOI: 10.1002/smr.2377

SPECIAL ISSUE - EMPIRICAL PAPER

Journal of Software: Evolution and Process WILEY

Design thinking and creativity of colocated versus globally distributed software developers

Rodi Jolak[1] | Andreas Wortmann[2] | Grischa Liebel[3] | Eric Umuhoza[4] | Michel R. V. Chaudron[5]

[1]Department of Computer Science and Engineering, Chalmers | Gothenburg University, Gothenburg, Sweden

[2]Department of Computer Science, RWTH Aachen University, Aachen, Germany

[3]School of Technology, Reykjavik University, Reykjavik, Iceland

[4]Department of Electrical and Computer Engineering, Carnegie Mellon University Africa, Kigali, Rwanda

[5]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, Netherlands

Correspondence
Rodi Jolak, Department of Computer Science and Engineering, Chalmers | Gothenburg University, Sweden.
Email: rodi.jolak@cse.gu.se

Present address
Rodi Jolak, Department of Computer Science, Chalmers | Gothenburg University, 41296 Gothenburg, Sweden

Abstract

Designing software is an activity in which software developers think and make design decisions that shape the structure and behavior of software products. Designing software is one of the least understood software engineering activities. In a collaborative design setting, various types of distances can lead to challenges and effects that potentially affect how software is designed. To contribute to a better understanding of collaborative software design, we investigate how communication gaps caused by social and geographic distances affect its design thinking and the creativity of its discussions. To this end, we conducted a multiple-case study exploring the design thinking and creativity of colocated and distributed software developers in a collaborative design setting. Compared with colocated developers, distributed developers spend less time on exploring the problem space, which could be related to different sociotechnical challenges, such as lack of awareness and common understanding. Distributed development does not seem to affect the creativity of their activities. Developers engaging in collaborative design need to be aware that problem space exploration is reduced in a distributed setting. Unless distributed teams take compensatory measures, this could adversely affect the development. Regarding the effect distance has on creativity, our results are inconclusive and further studies are needed.

KEYWORDS
cognitive aspects, collaborative design thinking, creativity, distance, empirical study, software engineering

1 | INTRODUCTION

When designing software, developers, together with other stakeholders, explore the interplay of problem and solution space. That is, they creatively ponder, make, and refine decisions that ultimately shape the final structure and behavior of the software product.[1]

Throughout the software engineering (SE) life-cycle, developers with various roles[2] jointly are *designing* the system. For example, developers do not only elicit requirements; they actually design the requirements by discussing and shaping these requirements with the contributing

Abbreviations: APIs, Application Programming Interfaces; GSE, Global Software Engineering; SE, Software engineering; UI, User interface.

stakeholders. Developers also design their code by modularizing, composing, analyzing, and evaluating source code. Similarly, developers design use cases, user interfaces, Application Programming Interfaces (APIs), and test cases. All of these activities demand creativity. To handle complex problems, expert developers intuitively practice *design thinking*,[3] which is a cognitive style, a mindset that helps developers in problem solving. In *design thinking*, developers explore the problem and solution spaces separately, and iteratively align the two. This process happens even if developers are not specifically trained in *design thinking*. While thinking about the design, developers might have sudden events of insight, leading to the establishment of key design concepts or decisions. These events are considered as indicators of creativity in the design process.[4]

Globally distributed projects are becoming the norm in SE,[5] and they raise social, technical, and organizational challenges.[6] Social challenges mainly involve communication challenges. In particular, globally distributed teams have little or no face-to-face communication, differences in time zones hinders synchronous communication, and language and cultural differences often lead to misunderstandings and lack of trust.[7,8] Technical challenges entail issues with shared instruments and resources, communication media, and collaboration tools.[9] Organizational challenges entail difficulties in work synchronization, coordination, and management of development teams between geographically distributed sites.[10] For instance, different engineering management processes between sites can cause problems to the integration phase.[7]

Consequently, it is quite likely that software design activities are affected by these challenges as well. In particular, it is unclear to what extent the iterative cycle of *design thinking* and the phenomenon of *design creativity* are hampered by the distribution of collaborating teams.

While the focus in research has been on technical artifacts of design (i.e., design notations and tools), there is only little work investigating design practices and cognition.[11] This focus on technical aspects of design is problematic, as SE is a sociotechnical endeavor, and further research controlling for human and social aspects is crucial for ensuring a successful engineering of software systems.[12] In fact, Petre and Van Der Hoek[13] argue that designing software is one of the least understood activities in which software developers engage.

To close this gap, we conducted an exploratory multiple-case study, investigating (1) how design *thinking* and *design creativity* take place during software design in colocated and in distributed teams and (2) how remote collaboration affects the occurrence of *creative events* between the participants. To this end, we replicate a design study that was originally done by Petre and Van der Hoek.[13] This original study consisted of the recording and analysis of a design session that was held with multiple designers at a single location. In our replication, we used the same design assignment but asked designers that were distributed across geographically different locations to collaborate on creating a design. For this, the distributed developers used a tool that enables sketching in real time and videoconferencing. We then analyze the differences in between these design processes. In particular, we qualitatively analyze how software developers switch between the problem and solution space of *design thinking*, how alignment takes place, and how that affects the occurrence of *creative events*.[4] Doing so, we aim to answer the following research questions.

- **RQ1** How does distance affect the design thinking of software developers?
- **RQ2** What challenges are encountered when collaboratively designing software at a distance?
- **RQ3** How does distance affect the design creativity of software developers?

Hence, the contributions of this paper are as follows:

1. We analyze the *design* process of software developers, leading to a better understanding of software design activities.
2. We qualitatively compare how *design thinking* differs between colocated and distributed setups, leading to an increased understanding of the effects of global SE on *design thinking*.
3. We analyze difficulties to distributed *design thinking*, thus exposing areas of improvement in global SE projects.
4. We qualitatively analyze the conversations between the participants to identify the occurrence of creative events in the locally and remotely collaborating teams as well as the discussions leading to these, which yields insights into the effects of remote collaboration on design creativity.

This paper extends our findings reported in Jolak et al.[14] Specifically, we added a novel qualitative analysis on design creativity, as well as the corresponding introduction and related work on this topic. Additionally, we updated the discussion accordingly to reflect new insights.

In summary, we find that distributed teams practice less *design thinking* compared with colocated teams. Furthermore, distributed teams focus more on solution space exploration and less on problem space exploration. Our results indicate that a lack of awareness and common understanding between the remote collaborators cause this change in *design thinking*. However, this change of focus in design thinking does not seem to affect the occurrence of creative events.

The remainder of this paper is organized as follows: We discuss related work in Section 2, followed by a description of the multiple-case study design in Section 3. We present the results in Section 4, followed by a discussion in Section 5 and the threats to validity in Section 6. We conclude and outline future work in Section 7.

2 | RELATED WORK

The results presented in this paper are related to *Design Thinking*, *Design Creativity*, and *Global Software Engineering* (GSE), which are highlighted in the following.

2.1 | Design thinking

According to Kimbell,[15] design thinking can be described as a cognitive style,[3,4] a general theory of design,[16] or an organizational resource.[17] One school of thought considers *design thinking* as an activity that the subject is aware of.[18,19] In contrast, several authors understand *design thinking* as a theory that explains how subjects practice problem solving during a design task without necessarily being aware of it.[3,4,16] In this study, we consider the latter understanding of *design thinking*.

Lindberg et al[20] highlight that *design thinking* fosters three main activities (see Figure 1):

1. *Exploration of the problem space*: by analyzing the problem space and framing the design problem;
2. *Exploration of the solution space*: by creatively devising and evaluating design solutions; and
3. *Iterative alignment of both spaces*: by keeping the problem space in mind for refining and revising the chosen solutions.

Furthermore, Lindberg et al[20] indicate that *design thinking* can broaden the problem understanding and problem solving capabilities in IT development processes. This is in line with Brooks,[21] who considers *design thinking* an exciting new paradigm for dealing with problems in software and IT development. Similar to Lindberg et al.,[20] Dorst and Cross[4] find that a designer's understanding of the problem and solution space co-evolve in an iterative fashion, until the designer finds a *bridge* that links concepts in the two spaces. The authors write that "creative design involves a period of exploration in which problem and solution spaces are evolving and are unstable until (temporarily) fixed by an emergent bridge which identifies a problem–solution pairing. A creative event occurs as the moment of insight at which a problem-solution pair is framed."[4] As a part of our study, we analyze these creative events in depth.

Petre et al[19] consider *design* as a goal-driven activity to decide upon a plan for a novel change in a specific context. This change, when realized, satisfies the contributing stakeholders. They underline that design thinking is conducted in different social contexts and at all stages of software development. Moreover, the authors claim that developers with a 'design-thinking mindset' perform contrasting design dialogs between problem and solution spaces, pragmatism and fitness-for-purpose, and across different levels of development focus and design cycles such as analysis, synthesis, and evaluation. Each of these dialogs provides a focus for design reasoning which helps to understand problems, manage complexity, and achieve enduring development success.

Dobrigkeit and de Paula[18] investigate how design thinking can support software development and how it manifests itself during the development process. By conducting a case study and interviews in a global IT company, Dobrigkeit and de Paula find that once trained in design thinking, developers find various ways to implement it throughout their projects even applying it to aspects of their surroundings such as the development process, team spaces and team work.

While, to the best of our knowledge, there are no studies that empirically assess the *design thinking* phenomenon in SE, there are few studies focusing on the social and cognitive activities of developers when they collaboratively practice problem-solving activities. Jolak et al[22] study how distributed software designers communicate and collaboratively make design decisions. They found that distance affects the quality of

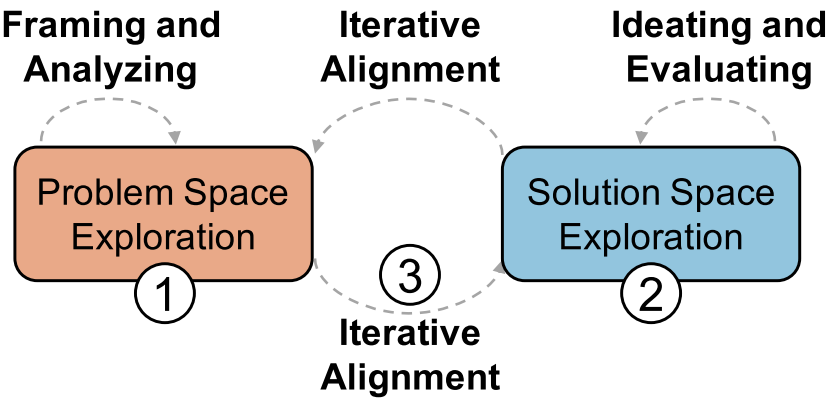

FIGURE 1 Problem solving with design thinking[20]

communication and reduces the amount of design decisions that the developers take during their distributed collaboration. Our multiple-case study uses the same data used in Jolak et al.[22] In contrast, in this study we focus on analyzing *design thinking* and challenges to distributed collaboration.

Similarly, Christiaans and Almendra[23] study how developers take decisions in software design. They find that developers in general tend to prioritize their previous knowledge in problem solving, while neglecting a thorough analysis of information in order to take decisions along their processes.

For understanding the software design process, Baker and Van der Hoek[24] analyze how developers generate ideas, discuss subjects, and do design cycles (i.e., exploring a design aspect, make some progress, and then switch to a new design aspect). They observe that the design process is highly incremental and developers repeatedly return to previously stated ideas and discussed subjects. This observation is in line with how expert developers behave when practicing *design thinking*, that is, by exploring the problem space, the solution space, and then repeatedly jumping back and forth to align the two spaces.

Razavian et al[25] consider software design as a problem solving exercise. They theorize that software design thinking requires two minds: a *reasoning* mind that focuses on the process of logical design reasoning and a *reflective thinking* mind that challenges the reasoning mind by asking reflective questions. Razavian et al. conduct multiple case studies to understand how reflections on reasoning and judgments influence software design thinking. They find that reflection improves the quality of software design discourse which, in turn, is considered as a foundation for a good design.[26]

In summary, *design thinking* is a cognitive style that fosters problem and solution space exploration and alignment between the two spaces through so-called "bridges." In software design, developers follow a process that is similar to *design thinking*, thus making the combination of the two, software design and *design thinking*, a relevant topic for investigation.

2.2 | Creativity in design

Dorst and Cross[4] characterize creativity in the design process as a phenomenon that occurs when designers have sudden, significant insights leading to the emergence of a key design concept or decision. They consider studies of creativity in design as necessary to develop a better understanding of how creative design occurs. Accordingly, they conduct protocol studies of nine experienced designers working on a small design assignment in order to observe creativity in the design. Activities such as defining and framing of design problem, ideation of original decisions, and co-evolution of the problem space and solution space (driven by surprising information) are reported to be key aspects of creativity. Moreover, different factors are observed to influence the design creativity, such as designer's own design goals and available design time.

Kruger and Cross[27] analyze different cognitive design strategies employed by designers and relate these strategies to design quality and creativity. The different analyzed design strategies are as follows: problem-driven, information-driven, solution-driven, and knowledge-driven design strategies. The results indicate that designers using a solution-driven design strategy tend to have a higher creativity but lower overall solution quality. Furthermore, designers using a problem-driven design strategy tend to have the best results in both design quality and creativity.

Wiltschnig et al[28] examine real-world design data from team-based design and development meetings. They analyze the design co-evolution within a collaborative context where several designs collaboratively create a design solution and examine whether the design problem and solution co-evolution concept captures collaborative creativity. The results show that design co-evolution iterative episodes are related with creative activities, such as analogizing and mental simulation which are critical for overcoming moments of uncertainty and for facilitating problem understanding and solution generation.[29] Hence, the results support the view that design co-evolution is a driver of creativity in collaborative design.

Creativity involves both divergent and convergent thinking.[30] Divergent thinking moves the thought away to consider different aspects that can foster new ideas and creative solutions. Convergent thinking brings together information and knowledge in order to solve problems. Maiden and Robertson[31] explore the integration of creativity as well as divergent and convergent thinking into the requirement engineering process. In particular, they study the effectiveness of a number of techniques for supporting the types creativity as identified by Boden:[32] exploratory, combinational, and transformational creativity. Exploratory creativity enables seeing possibilities and investigating new idea. Combinational creativity enables the alignment of ideas and requires the ability to find links between concepts. Transformational creativity involves the transformation of some dimensions of the thinking space, so that new ideas can be generated. The results of Maiden and Robertson show that brainstorming is more effective than analogical reasoning for exploratory creativity. Storyboard development is found effective for combinational creativity. Moreover, results revealed that removing constraints leads to the generation of more ideas and thus supporting transformational creativity.

Mohanani et al[33] study the effect of presenting desiderata as ideas, requirements, or prioritized requirements on design creativity. They find that using desiderata framed as requirements or prioritized requirements lead to the creation of designs that are less original but more practical than the designs created by using desiderata framed as ideas. Accordingly, it is suggested that more formal and structured presentations of desiderata are less appropriate where a creative solution is desired.

Nguyen and Shanks[34] provide a framework for understanding creativity in requirement engineering. They state that a creative product is characterized by being novel, original, and useful. Moreover, they state that surprisingness is often associated with creative products. Considering

creative processes, an inspirationalist view is discussed. According to this view, a creative process is characterized by moments of preparation, incubation, illumination, and verification and expression of insights. These insights occur at the "Aha!" moment, when a long-sought idea or solution suddenly appears at the conscious level.

2.3 | Global software engineering

GSE is the practice of engineering software systems across geographical, sociocultural, and temporal boundaries.[8] Software organizations opt for globalizing their projects mainly to maximize business profits by taking advantage of low development cost and time, achieving a high percentage of productivity, accessing a skillful workforce, and using innovative concepts.[35] Also, in 2020 the COVID-19 epidemic has forced many software development teams to work in a distributed manner—even while their team-members still live in the same city. However, these organizations often face numerous challenges, including poor quality of globally developed software.[36]

Awareness and common understanding are primordial for distributed collaboration. According to Dourish and Bellotti,[37] awareness is the understanding of the activities of others. Gutwin et al[38] describe four main types of awareness:

- Work-space awareness: knowing and understanding the interactions of others with the work-space and its artifacts.
- Informal awareness: sensing who is around and what are their intentions.
- Group-structural awareness: knowing the roles and responsibilities of the collaborators, positions on issues, and team's process.
- Social awareness: the knowledge that a person gains about others in a social context, such as level of interest and emotional state.

According to Clark,[39] common ground or understanding is the knowledge that collaborators are aware of and have in common. In other words, a team achieves a common ground or understanding of a concept if the members of the team know the topic and they all know that they all know the topic. A lack of common ground or understanding influences communication effectiveness as well as the activities and results of the collaborators.[40]

Herbsleb[8] studies the impact of geographic distance on distributed collaboration. He suggests that colocation facilitates communication since software developers explicitly know who is working and what is happening in the working place. The suggestion of Herbsleb is in line with Damian et al,[41] who observe that geographic distance hampers awareness of remote collaborating teams participating in GSE.

Besides geographic distances, communication gaps can be caused by other distances. Bjarnason et al[42] present a theory on different distances and their influence on communication and coordination in software development projects. The authors suggest that some of these distances can be shortened by following certain practices, such as

- involving roles from different disciplines to perform an SE activity,
- reviewing documentation and artifacts, or
- performing incremental SE.

Of particular interest to our study are communication gaps caused by social distances, since these distances are often amplified by geographic distance and since they cannot easily be mitigated by technological solutions. Geographic distribution, for example, due to outsourcing, often leads to a more diverse mix of cultures which can give rise to communication gaps. For instance, Lehmann-Willenbrock et al[43] observe that culture can introduce differences in the behavior of distributed collaborating teams. Moreover, as part of a substantial body of work on culture, the Hofstede culture dimensions[44] are a well-known theory aiming to explain how different cross-local practices developed in families, schools, and organizations introduce differences in thinking and social actions. Even though Hofstede's dimensions help to identify that culture has an effect, they do not help in understanding the interplay between culture and the practice, which limits the understanding of using culture to explain social challenges.[45] Hence, using culture to explain social challenges with a limited understanding of its impact might, in turn, close down any further investigation of the actual and situated challenges of bringing different work practices together across distances.[46]

In addition to the culture dimension, Bjørn et al[47] find that geographic distance raises more social challenges which are considered critical obstacles for successful remote collaboration. Overall, we see that more in-depth studies of the activities and behavior of distributed developers are needed.[48]This should result in a better understanding of how to account for technological and social challenges caused by geographic distance.

The design assignment that we use in our study is intentionally formulated to trigger design thinking and reasoning. Informal notations and whiteboards are often used during collaborative design ideation and reasoning to explore problems and externalize design solutions.[49] Whiteboards do not constrain the modeling notation that can be used and, thus, support informal modeling and design thinking. Therefore, in our study, we use whiteboards (standard whiteboards in the colocated case and interactive whiteboards in the distributed case) to better study the design thinking phenomenon.

Dekel[50] study colocated software design meetings in order to observe the activities of developers and outline requirements for tools supporting distributed software design. For distributed software design, Dekel suggests that the design tools should mainly support the creation of informal notations to capture ideas while brainstorming. There are several tools that support distributed software design and the creation of informal notations, such as the Software Design Board,[51] Calico,[52] Metaglue,[53] and OctoUML[54] (a tool that we developed). In this study, we use interactive whiteboards with a simplified version of OctoUML to support and explore distributed software design. More details are provided in the next section.

3 | CASE STUDY DESIGN

The intention of this study is to explore the *design thinking* and *design creativity* of colocated and distributed software developers. Moreover, we seek to identify challenges and impediments that could hinder collaborative distributed *design thinking*. We aim to seek insights and generate hypotheses for future research by observing the process and outcome of the *design thinking* and *creativity* phenomena. But as it is hard to study *design thinking* and *creativity* in isolation and separate it from its context, we chose a case study design, where the boundary between the phenomenon and its real-life context cannot be clearly specified.[55] Our case study is an exploratory and inductive multiple-case study.[56]

3.1 | Cases and units of analysis

Figure 2 shows the design of our multiple-case study, which serves to examine two cases:

- **Case 1** (*Colocation*): Collaborative colocated designing of software architecture using a whiteboard. This is a single case[55] with one Unit of Analysis (UoA): *Design process*. Here, we analyze the problem-solving cognitive style of developers and, in particular, focus on the *Design thinking* and *Design Creativity* phenomena.
- **Case 2** (*Distribution*): Collaborative distributed designing of software architecture using an interactive whiteboard. This is an embedded case[55] with two UoA: *Design Process* and *GSE Challenges*. In this case, we analyze the challenges that hinder collaborative distributed designing.

3.2 | Theoretical framework

While our case study is exploratory in nature, even this kind of case study should have a foundation consisting of theories, if possible, and propositions/hypotheses.[55,56] Next, we detail the theoretical framework that we employ in this multiple-case study.

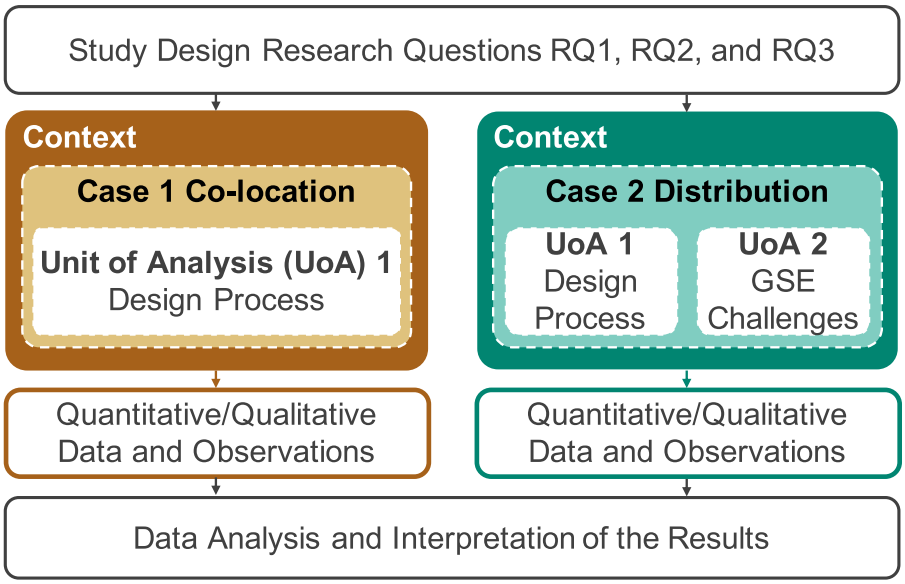

FIGURE 2 Multiple-case study design featuring the two use cases of colocated and distributed software design

In this study, we analyze the software design process and reason about how individuals solve problems. First, we make use of the *design thinking* phenomenon and analyze the interactions of individuals to the three *design thinking* activities described by Lindberg et al.:[20] exploration of the problem space, of the solution space, and iterative alignment between the two. While developers can be explicitly trained in *design thinking*, we follow the line of research that assumes that even untrained developers perform *design thinking*. Second, we analyze the design discussions to explore how creative design occurs, that is, understand the events, activities, or behaviors that contribute to design creativity.[4] The two cases are selected to predict possible contrasting results on the *design process* by altering one condition: the geographic distance (colocation vs. distribution).

In addition to the *design thinking* and *creativity* phenomena, we analyze our data with respect to Bjarnason's theory on distances in SE.[42] In particular, we study communication gaps caused by social distances, since these distances are often amplified by geographic distance and since they cannot easily be mitigated by technological solutions. Using these distances, we aim to reason about the differences between the two cases, as well as the perceived challenges of being distributed (UoA 2 in Case 2).

The related work on GSE, discussed in Section 2, shows that remote collaboration can hinder effective communication. We relate this overall conclusion to our RQs by making the following three propositions, which then guide our case study design:

- **Proposition A1**: We propose that communication gaps caused by social and geographic distances do not affect *design thinking*. *If* the *design thinking* of the two cases varies considerably, *then* this indicates that these distances affect *design thinking*. This would encourage deeper investigations of the effect of these distances on *design thinking*.
- **Proposition A2**: We propose that communication gaps caused by social and geographic distances do not affect *design creativity*. *If* the *design creativity* of the two cases varies considerably, *then* this indicates that these distances affect *design creativity*. This would encourage deeper investigations of the effect of these distances on *design creativity*.
- **Proposition B**: We propose, based on the findings in previous studies,[8,57] that poor GSE tool support and difference imposed by social factors are the most frequently reported challenges that affect GSE. *If* the perceived challenges in our study differ from those reported in related work, *then* this indicates that additional factors in our study setup influence the perceived challenges. Moreover, this would encourage further investigation of the perceived challenges and confounding factors.

To investigate Propositions A1 and A2, we study whether and how communication gaps caused by social and geographic distances do influence design thinking and creativity. To investigate Proposition B, we explore different factors imposed by social and geographic distances that contribute to such possible communication gaps.

3.3 | Context

For both cases, we used a software design challenge, in which designers are asked to design the signal timing at a road intersection with the goal that traffic is flowing in a fluid manner and waiting times are minimized.[13] The challenge documentation is available online[1]. The designers are asked to create the architecture of a simulator enabling its users to investigate the effect of different signal timing on the traffic flow. The challenge is of realistic size and complexity and focuses on four functional requirements:

1. Users can create a visual map of intersected roads of varying length.
2. 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.
3. Users can simulate traffic flows on the map, and the resulting traffic levels are conveyed visually.
4. Users can change the traffic density per road.

Prior to starting the challenge, developers were informed that:

- their design will 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.

Additionally, the developers signed a consent form. No compensation was offered, apart from a snack at the Swedish site. Ethics approval was not needed and therefore not obtained.

[1]Challenge documentation: (https://www.ics.uci.edu/design-workshop/files/UCI_Design_Workshop_Prompt.pdf).

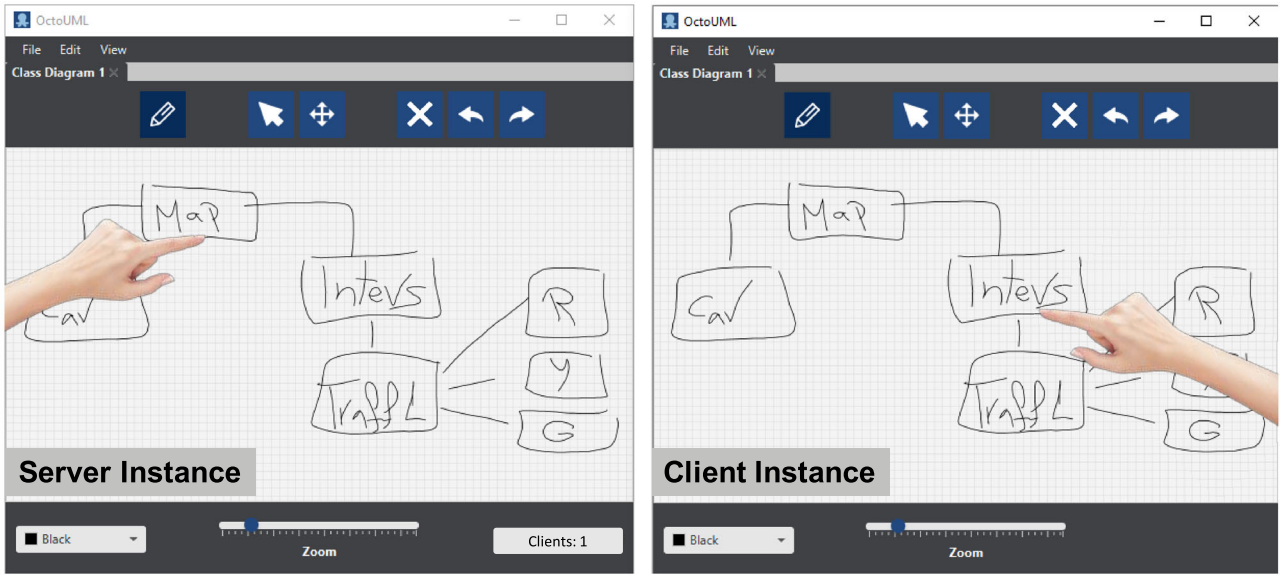

FIGURE 3 The main views of the server and client instances of OctoUML

Developers were given a printed copy of the design challenge and had a maximum of 2 h to work on their design. The developers were given no additional instructions, other than to use the whiteboard for any writing or drawing that they wished to perform. After completing their work, the developers were given 10 min to collect their thoughts and briefly explain the design.

For Case 1, we used the data set provided by Petre and Van der Hoek,[13] who performed the study with three (colocated) teams of two professional software developers each. The developers in these teams were considered expert designers (i.e., their companies would trust them in solving key software design problems) and had on average 19 years of experience. We do not know whether they had previous experience or training in *design thinking*.

For Case 2, we recruited three teams of two professional software developers to work on the same design challenge, but from two different geographic locations: Aachen, Germany, and Gothenburg, Sweden. The developers were recruited through personal contacts and were paired based on availability. The pairs had not worked together prior to the study. The three distributed pairs had 20 and 5 years (Distributed Team 1, DT1 hereafter), 6 and 7 years (DT2), and 5 and 5 years (DT3) of general software engineering experience. None of the developers in Case 2 had previous experience or training in *design thinking*.

Instead of a regular whiteboard, we used interactive whiteboards with a simplified version of OctoUML[2], an open source software design environment[54] supporting remote collaborative design sessions between geographically distributed developers. To collaborate remotely, one developer creates a *server* instance of OctoUML. Other remote developers can connect as *client*s using TCP/IP protocol. Figure 3 shows the main views of the server and client instances of OctoUML. Developers can simultaneously sketch on the shared canvas of OctoUML using special styluses or by using their fingers, as interactive whiteboards support touch-input. While OctoUML has some UML capabilities, like creating class shapes, we removed those in the study to make OctoUML resemble a regular whiteboard as closely as possible and, thus, mitigate the impact of other factors than the geographic distance on the UoA 1 (i.e., design process) of the two cases. Similarly, choosing another commercial remote collaboration tool might have introduced other collaboration features that would have changed the experience compared with a regular whiteboard. Hence, we opted to choose OctoUML since we could customize it for the study's purpose. The interactive whiteboards were connected to computers providing video conferencing (via Skype[3]) between the two locations.

Immediately after each distributed design session, we asked the developers to evaluate the usability of OctoUML. The reason is to understand to which extent the usability of OctoUML did impact the work of the distributed developers. In particular, we asked the developers to answer the System Usability Scale (SUS) questionnaire. The SUS is an easy, standard way of evaluating the usability of a system.[58] It is a form containing 10 statements, and users provide their feedback on a 5-point scale (1 is "strongly disagree" and 5 is "strongly agree"). SUS effectively differentiates between usable and unusable systems by giving a measure of the perceived usability of a system. It can be used on small sample sizes and be fairly confident of getting a good usability assessment.[59] Figure 4 shows the results of SUS evaluation. Overall, we observe that the

[2]OctoUML Website (https://rodijolak.com/#octouml).
[3]Skype Website (https://www.skype.com/en).

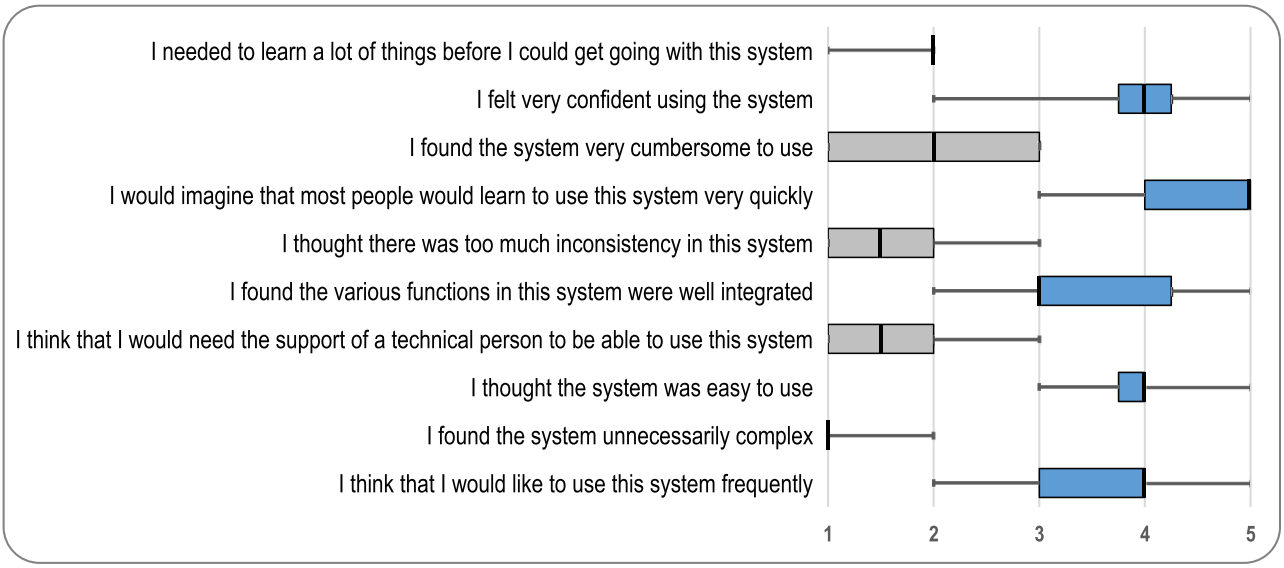

FIGURE 4 The perceived usability of OctoUML

perceptions regarding the usability of OctoUML were positive. Regarding the SUS score, OctoUML received an average SUS score of 74.17 ± 5.63, which can be interpreted as a grade of *B*− (i.e., a good usability score) according to Sauro.[60]

3.4 | Data collection and analysis

In both cases, design sessions were recorded with a video camera that was positioned to capture the whiteboard and the developers at work. In Case 2, we recorded two videos per session, one each for the two geographic locations. We then used verbatim transcriptions for subsequent analysis. For Case 1, the transcriptions were available online[4]. For Case 2, the first two authors transcribed the recordings verbatim[5]. The resulting transcriptions were not reviewed.

3.4.1 | Design thinking

We analyzed the transcriptions of approximately 10 h of design activity by six pairs of professional software developers and performed a manual coding of more than 2000 conversation dialogs. We created a coding schema (see Figure 5) to capture design decisions from the problem space (traffic flow) and solution space (SE). This schema is based on the design-reasoning decisions of Weinreich et al.[61] Dialogs in the transcriptions were then assigned codes (using NVivo[6]). If multiple codes were assigned per dialog, we ordered them by the time they occurred in the transcription. The coding schema is described in detail, and examples are given in Appendix A.

The first two authors, who had 4 and 2 years of experience with qualitative analysis, performed the analysis for this aspect. To ensure reliability, we followed established advice on selecting and reporting intraclass correlation coefficients (ICCs).[62]

We performed two-way mixed ICC (3,k) tests with 95% confidence interval on 11% of the data. The ICC value is 0.84, which is considered a good reliability.[62] After that, the two coders discussed and aligned the differences in their coding. Finally, the two coders collaboratively continued to code the rest of the data, that is, 89% of the data. Based on the frequency of occurring codes and their order, we derive the *design thinking* graphs reported and discussed in Section 4. Essentially, we derive the frequency of each *design thinking* phase by counting (see Figure 5 as a reference):

- how frequent the developers explored the problem space (Codes 1–5),
- how frequent the developers explored the solution space (Codes 6–16), and
- how frequent the developers iteratively align the two spaces (subsequent codes changing from problem to solution space, or vice versa).

[4]Case 1 Transcriptions: (https://www.ics.uci.edu/design-workshop/videos.html).
[5]Case 2 Transcriptions: please send a request to rodi.jolak@cse.gu.se
[6]NVivo Website (https://www.qsrinternational.com/nvivo).

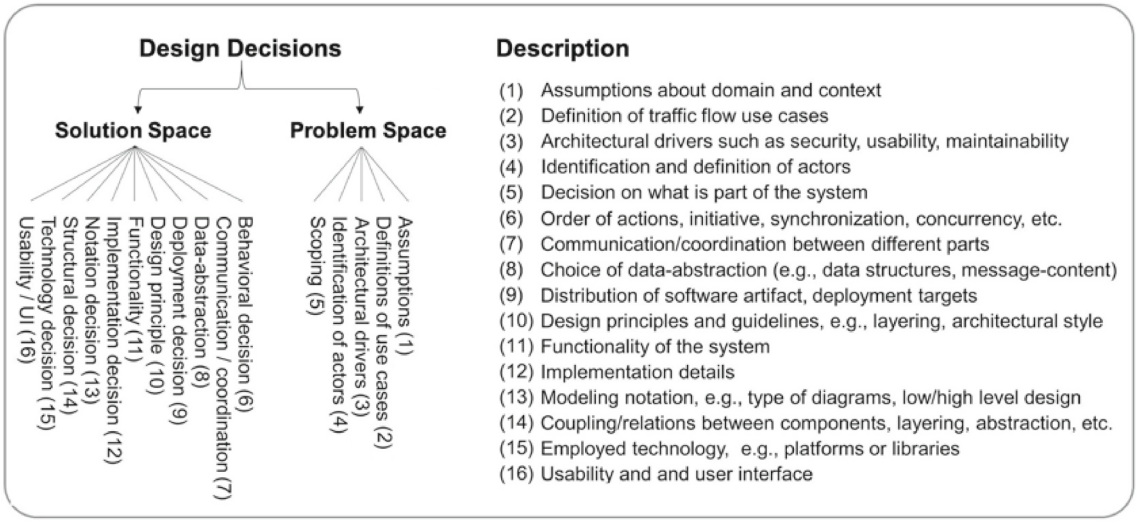

FIGURE 5 Classification schema of design decisions into problem space and solution space decisions (based on Jolak et al.[22])

3.4.2 | Design creativity

In addition to this quantitative summary of the codes, we performed a qualitative analysis of *creative events*[4] in the interactions between each pair of designers. A creative event in *design* is the "moment of insight" at which "an emergent bridge which identifies a problem–solution pairing [..] is framed."[4] Similarly, Nguyen and Shanks[34] describe moments of insights or "Aha!" moments.

Four of the authors participated in the analysis process. Each transcript was analyzed by one of the authors (with 4 years, 2 years, and 0 year of experience in qualitative analysis). Additionally, as the author with the longest experience (8 years) in qualitative analysis, the third author analyzed all transcripts and discussed the resulting coding with the remaining authors. We followed this process to discuss and clarify differences in coding and to ensure a consistent coding approach.

First, we tried to identify creative events based on the following descriptions derived from Dorst and Cross:[4]

1. Subjects understand a connection that they were unaware of beforehand. This connection simplifies an issue they were dealing with so far, resulting in a solution to a part of the assignment/task. OR
2. Subjects identify an issue they were unaware of before, leading to a complication of the overall task. OR
3. Subjects find an appealing simple solution to an issue. This greatly simplifies the issue or the overall task. OR
4. In terms of keywords, the subjects show sudden realization (e.g., "Aha," "Now I understand it," "Wait a moment, we haven't thought about…"). OR
5. The subjects feel that they have understood or uncovered the underlying problem behind or the difficulty with the task.

Keywords alone do not clearly identify a creative event, as they are context-dependent (e.g., a matter of fact "Now I understand what you are saying" vs. an emotional "Now I finally understood the assignment"). Therefore, this task is to some extent subjective. Dorst and Cross however state that creative events are generally "highly emotional steps."[4] Since we did not find any events that would correspond to this strict classification of creative events as "highly emotional step," we relaxed the coding guidelines to also include general realizations that were not emotional in nature. This is further discussed in Section 5.3.

After identifying all creative events, the authors processing the transcripts coded the discussion before and after the creative event using an open, descriptive coding approach.[63] We then used the resulting high-level codes to summarize how the subjects reached the creative events and what distinguished different approaches. Afterward, we assigned two labels to each event. First, we identified whether the creative event was concerned with the problem space, the solution space, or a bridge between the two spaces. This links our analysis of creative events to the *design thinking* phenomenon and allows us to compare and discuss the findings of both analyses. Second, we classified whether the discussion before reaching the creative event was convergent or divergent in nature. The presence of a combination of convergent and divergent discussions is believed to be an indicator of design creativity.[64] The two discussion modes are related to two thinking modes from a neurological point of view.[30] Divergent discussions are characterized by "defocused attention."[30] We coded discussions as divergent if the topic being discussed shifted multiple times in the time leading up to a creative event. In contrast, convergent discussions are characterized by "focused attention,"[30] which we assigned to discussions that essentially focused on a single topic or concept.

We plotted the resulting creative events with their associated labels on a time line and discussed differences in frequencies, location, and labeling of creative events between different teams.

3.4.3 | Challenges to distributed design

In each geographically distributed site (Sweden/Germany), one supervisor attended the design sessions to observe the design process and note observed challenges. In addition, after each distributed design session, we collected data through a poststudy questionnaire. We asked the developers to indicate and elaborate on eventual challenges to their distributed collaborative design experience via an online form (self-evaluations) using the following two open questions:

- Q1. What was challenging in this experience and what was missing in your opinion?
- Q2. Did you recognize challenges due to being geographically distributed? If you did, which challenges did you find?

Collecting data on GSE challenges from two sources (perceptions of the distributed developers and our observations of the design sessions) allowed us to triangulate the data on the experienced challenges.

4 | RESULTS

In this section we present the results of this study based on our three RQs. A detailed discussion of those results is provided in Section 5.

4.1 | RQ1: Colocated versus distributed design thinking

Figures 6 and 7 illustrate the *design thinking* processes of the colocated and distributed teams, respectively. The numbers in the figures represent the absolute and relative frequencies of (i) problem space exploration, (ii) solution space exploration, and (iii) alignment of the two spaces that are practiced by each team in Cases 1 and 2.

Considering absolute frequencies, we observe a higher number of *design thinking* interactions in the colocated teams (CT1: 303, CT2: 203, CT3: 323 interactions) than in the distributed teams (DT1: 140, DT2: 179, DT3: 122 interactions). Furthermore, we notice that the teams in Case 1 did more problem space exploration and more alignment between the problem space and solution space than the teams in Case 2.

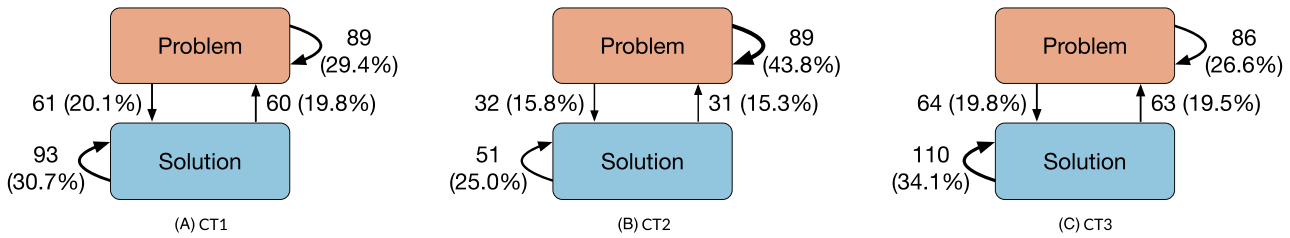

FIGURE 6 The *design thinking* of colocated teams CT1, CT2, and CT3 (Case 1))

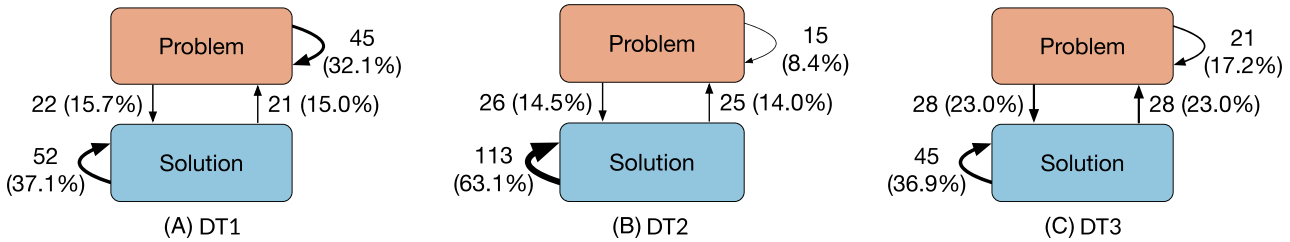

FIGURE 7 The *design thinking* of the distributed teams DT1, DT2, and DT3 (Case 2)

In terms of relative frequency, we see that all teams in Case 2 have a larger percentage of solution space exploration than any of the teams in Case 1. Specifically, the two developers in DT2 were more solution oriented (i.e., 63.1% of their *design thinking* was into the solution space) and did more solution space exploration than any other team, colocated or distributed. Focusing only on Case 2, we notice that all teams explored the problem space (DT1: 32.1%, DT2: 8.4%, DT3: 17.2%) less than the solution space (DT1: 37.1%, DT2: 63.1%, DT3: 36.9%).

Overall, the results contradict Proposition A that design thinking is not affected by communication gaps caused by social and geographic distances. We summarize this as follows:

> **Observation 1.** Communication gaps caused by social and geographic distances affect design thinking by reducing problem space exploration and alignment between problem and solution space.

4.2 | RQ2: Challenges to distributed design

While RQ1 is concerned with how *design thinking* is affected by geographic distribution, RQ2 aims to answer why geographic distance affects *design thinking*, that is, what challenges designers perceive when working in a distributed fashion. In the following, we present which challenges were perceived by the distributed teams in Case 2.

Based on the poststudy questionnaire (see results in Figure 8) and our observations, we find that there are multiple challenges to designing in a distributed way.

All six participants in Case 2 found that it was challenging to be aware of the remote individual's reactions to interactions or the remote individual's focus on the joint work. We also clearly observed this challenge during the design sessions. For instance,

"Where do you draw?", *"I don't see what you draw now"*, and *"- I don't see you drawing at the moment. - I drew out the left of the map"*. Configuring the environment with more video cameras so that facial and hand gestures of developers can be seen independently of their position could mitigate this issue.

The second challenge mentioned by all participants in the posttask questionnaire was a lack of common understanding; the participants reported, for example, that "It is challenging to get a common understanding for the problem," that they are "missing personal communication and eye-contact," and that they "couldn't see how my partner reacted to proposals." The lack of work-space awareness could have contributed to the lack of common understanding. Indeed, not perceiving the actions of the collaborating partner influences contact initiation and often leads to misunderstanding of communication content and motivation.[8] To mitigate this challenge, some participants suggested leveraging more formal notations for communicating the design decisions, for example, "to improve the general understanding" and to "avoid misunderstandings."

Two participants mentioned network problems. We report quotes from the participants regarding these problems: *"There is no connection anymore"*, *"For some reason the Skype connection is getting worse"*, and *"You are no longer connected, we see clients zero"*. Furthermore, technology contributes strongly to the awareness challenge and could additionally contribute to misinterpretations of discussions (reported by one participant).

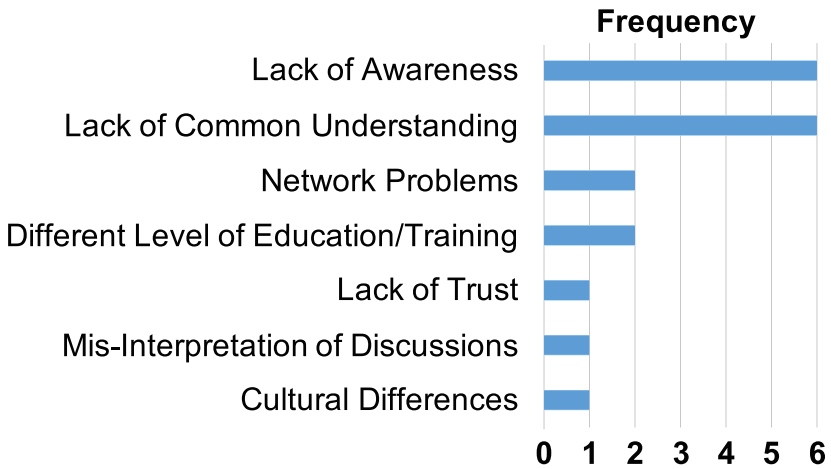

FIGURE 8 Perceived challenges to distributed design

Observation 2. Main perceived challenges to online distributed software design includes the following: lack of awareness, lack of common understanding, and network problems.

4.3 | RQ3: Creative events

We identify and report a number of *creative events* during the design sessions of both the colocated and distributed teams. We further analyze and discuss the dialogs or activities that lead to these creative events.

Tables 1 and 2 report the observed creative events for the colocated and distributed teams, respectively. The second column in each table shows the frequency of creative events that relate purely to the problem space, to the solution space, or connect the two. The third column then further breaks each type down into those events that were reached through divergent and those reached through convergent discussions.

4.3.1 | Colocated teams

The three colocated teams collaborated on a design task, spending between 60 and 110 min on the task. Their creative events and the discussions leading to these are illustrated in Figure 9. Here, each graph represents a conversation and structures it according to creative events (graphical peaks), discussions leading to creative events (elevated areas), and other discussion. The discussions leading to creative events can be either convergent or divergent.[30]

The conversations feature between 6 and 0 creative events relating to the problem space, between 2 and 7 creative events relating to problem-solution bridges, and between 2 and 18 creative events relating to the solution space. Out of these, between 6 and 20 discussions leading to these events are convergent and between 3 and 5 discussions are divergent.

Considering the different lengths of discussions relative to the number or kinds of different events, this hints that quality and creativity are independent of discussion length. Moreover, the three conversations yield very different structures: the occurrence of creative events begins after initial discussion of different duration; the kinds of creative events and discussions are different and their distribution is as well.

We report our observations on the discussions of the three teams below.

TABLE 1 Creative events of the colocated teams

Team	Event type (frequency)	Discussion (frequency)
CT1	Problem (0)	Convergent (0)
		Divergent (0)
	Problem–solution (7)	Convergent (7)
		Divergent (0)
	Solution (18)	Convergent (13)
		Divergent (5)
CT2	Problem (6)	Convergent (4)
		Divergent (2)
	Problem–solution (3)	Convergent (3)
		Divergent (0)
	Solution (2)	Convergent (1)
		Divergent (1)
CT3	Problem (0)	Convergent (0)
		Divergent (0)
	Problem–solution (2)	Convergent (1)
		Divergent (1)
	Solution (8)	Convergent (5)
		Divergent (3)

TABLE 2 Creative events of the distributed teams

Team	Event type (frequency)	Discussion (frequency)
DT1	Problem (4)	Convergent (2)
		Divergent (2)
	Problem–solution (6)	Convergent (3)
		Divergent (3)
	Solution (11)	Convergent (11)
		Divergent (0)
DT2	Problem (4)	Convergent (3)
		Divergent (1)
	Problem–solution (4)	Convergent (4)
		Divergent (0)
	Solution (5)	Convergent (5)
		Divergent (0)
DT3	Problem (0)	Convergent (0)
		Divergent (0)
	Problem–solution (4)	Convergent (2)
		Divergent (2)
	Solution (7)	Convergent (4)
		Divergent (3)

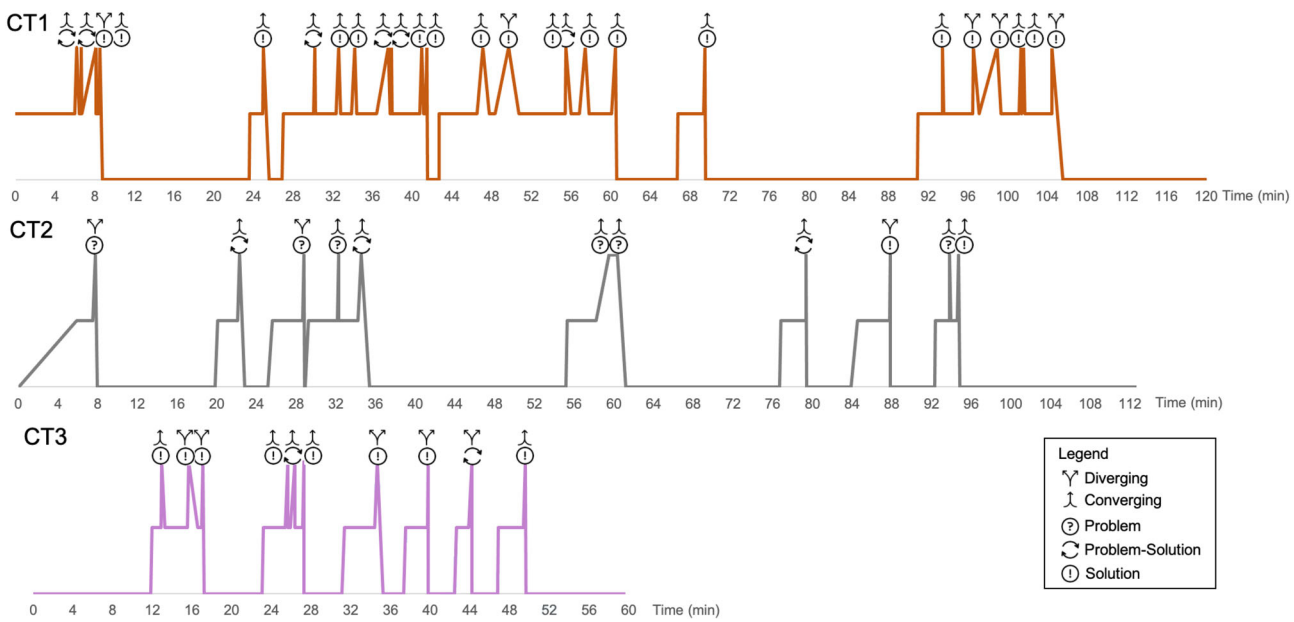

FIGURE 9 Timeline of of the three colocated teams presenting their creative events regarding problem space, solution space, and problem-solution bridges including discussions leading to these events. Icons refer to the spatially closest peak

Colocated team CT1

At the start of CT1's interaction, the team follows a pattern in which required concepts are enumerated in a quick order. The first two creative events are problem-solution bridges that are directly linked to the requirements, for example: *"Okay, so intersection, then we need car, we need the notion of time, this is a simulation of time, right."*

Overall, CT1 does not seem to have any *sudden* realizations or highly emotional moments. However, they engage in long dialogs about one single concept or issue, ultimately converging to a solution, a simplification, or an assumption: *"Yeah, right away rules." "Okay this is getting more complicated." "It's coming together though, I think. It's just more rules." "Yeah, yeah just need, well one probably also needs an admin right?"*. In terms of our codes, this is visible in the absence of creative events that relate purely to the solution space, in a high proportion of solution events (18 out of 25), and in a majority of convergent discussions (20 out of 25).

On several occasions, the team takes real-world examples into account to reflect on their solution and to ask themselves whether the solution needs to be more complex, for example: *"Do you want to keep it or do you want to do two queues?" "It depends if there's two lanes or not." "Lanes… Yeah, the lane would be an addition of the road with some specific logic data."*.

Colocated team CT2

CT2 discuss different aspects of the overall task at depth. However, at several occasions, when encountering a complex or unclear issue, they make an assumption or decide that they keep it as a question to the customer later on: *"This is something where I'd go back to the customer and try and figure out, how did they collect this data, you know like Professor E must have statistics about, you know San Francisco traffic"*. This enables the team to cover a large amount of topics during their session and also to a conclusion of many topics at a shallow level, without deeper discussions. Thus, the team hardly ever reaches any realizations.

Interestingly, CT2 has an event that comes closest to the definition by Dorst and Cross[4] towards the end of their design session. After several assumptions and discussions referred to the customer, they finally seem to realize the true goal of the task: *"Exactly, exactly. Because that's the purpose of Professor E's thing is to how do you correlate these settings with the result."*. This realization allows the team to reason better about several of the assumptions and simplifications made earlier on.

Another interesting observation is that CT2 has the highest proportion of creative events relating to problem space and problem–solution bridges. Indeed, only 2 creative events relate to the solution space, out of 11 overall.

Colocated team CT3

CT3 exhibits both phases of divergent and of convergent thinking. In some cases, they switch frequently between topics. For instance, in the beginning of their design session, the team quickly covers a lot of different topics, leading to a realization which part of the task is of major importance: *"So it seems like [..] designing intersections is kind of the key thing. I think so." "Yeah, I agree."* Later on, they re-visit the task in a structured way to see what parts are missing: *"Okay cool, so now come back up, what big concepts haven't we captured yet?"*. Creative events are distributed evenly throughout their interaction, compared with most other teams that tend to have creative events predominately during the beginning and towards the end of their interactions.

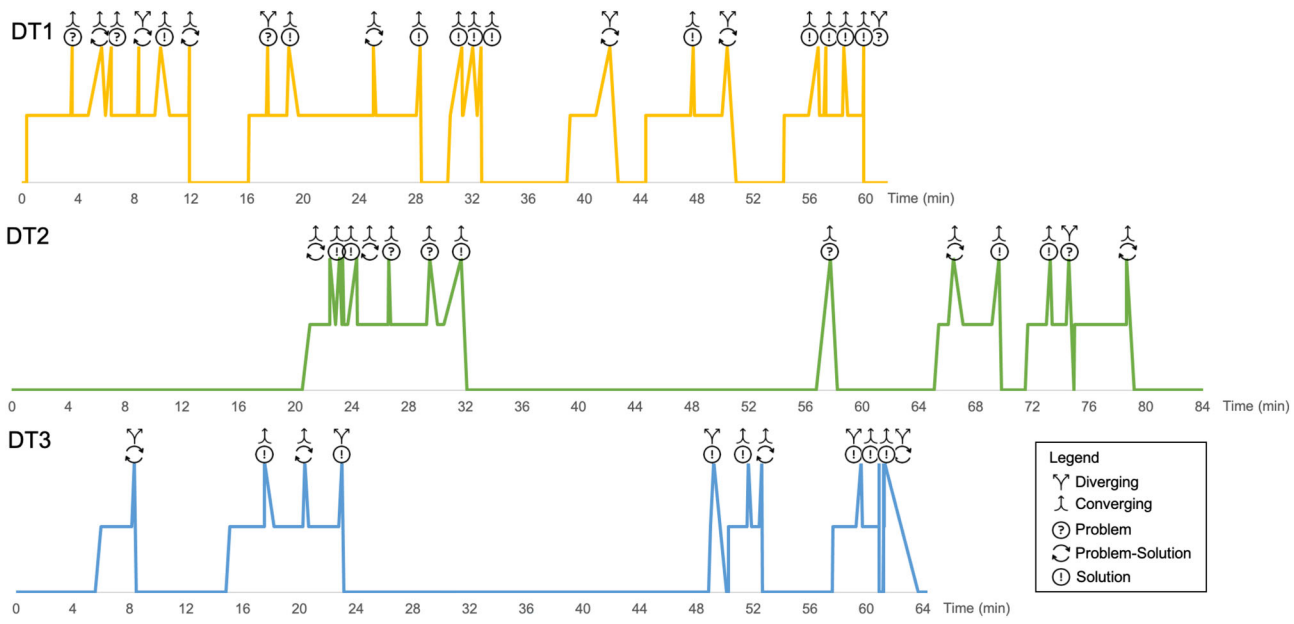

FIGURE 10 Timeline of the three distributed teams presenting their creative events regarding problem space, solution space, and problem–solution bridges including discussions (converging or diverging) leading to these events. Not all creative events are represented graphically due to space limitations. Icons refer to the spatially closest peak

4.3.2 | Distributed teams

The three distributed teams collaborated on a design task, spending between 73 and 90 min on the task. Their creative events and the discussions leading to these are illustrated in Figure 10. Again, each graph represents a conversation and structures it according to creative events (graphical peaks), discussions leading to creative events (elevated areas), and other discussion. The discussions, again, can be either convergent or divergent.[30]

The conversations yield between 0 and 4 creative events regarding the problem space, between 4 and 6 creative events regarding problem–solution bridges, and between 5 and 11 creative events regarding the solution space. The discussion leading to these events are mostly convergent (between 5 and 16 discussions) and less often divergent (between 1 and 5 discussions). From this graphical overview, we can see that our results confirm Proposition A2 that design creativity is not affected by communication gaps caused by social and geographic distances. However, there are large differences between teams. We summarize this observation as follows:

> **Observation 3.** Communication gaps caused by social and geographic distances do not clearly affect design creativity. Overall, there are large differences between teams in the frequency and location of creative events.

Below, we detail observations on the individual discussions.

Distributed team DT1

Developers in DT1 follow a strategy in which, initially, they iterate on what is required. Therefore, the first seven creative events are exclusively about what is required. *"But maybe first… so these are the things we need to do at real time.[..] " "Do you have something else? [..]" "I guess there should be some statistic [..]" "Is it mentioned in the task or do you just think it would be useful?"*

After the initial exploration, the remainder of the discussion is mainly solution-focused. Essentially, different requirements are addressed one by one or "ticked off." There is only very little problem focus during this period. Indeed, DT1 has only a single creative event that is purely focused on the problem space: In the very end of their dialog, they discuss whether any requirements are left. The discussion is predominately convergent. Divergent episodes are only found when the team tries to get an overview of the problem space (in the very end) or an overview of how concepts from the problem space are addressed in the solution.

Distributed team DT2

DT2 has structured discussions with little jumps between topics, mainly in a convergent fashion. The only divergent creative event occurs towards the end of the session, when the team discusses any remaining requirements: *"Is there anything else what we are missing right now, despite the sensors?"* Sometimes, realizations arise out of the blue but still seem to be connected to the ongoing (convergent) discussion. For instance, at a point where the team discusses the topic of how to simulate intersections, one team member realizes that sensors have not been covered: *"So do we, ah, ok what we are missing right now is the sensor"*.

In contrast to, for example, DT1, the team does not cover many parts of the problem domain, since they end up discussing the intersection and simulation at length. However, their discussions go to a much deeper level of the problem space, and fewer assumptions are made. This is also evident in the large proportion of creative events that are connected to the problem space or connected to bridges in the problem and solution spaces. A technique the team regularly uses is to think through a real-life case and try to compare that to the task at hand: *"At least here in Sweden [..] the main road is always green if it doesn't detect anything coming from the crossing roads. So the big road is always green, but when there are cars coming in from left or right it detects that and then it sets the main road to red so that they can join."*

Distributed team DT3

In DT3, several realizations in the beginning (after approximately 20 min) are essentially shortcut by making simplifying assumptions: *"I would say so, but I would do this on some… We can rely on third party libraries. So there is a library that can randomly compute how the cars would move."*. In particular, the assumption that simulation is covered by a third party library avoids a discussion that took the majority of time in other teams.

Several minor realizations occur through enumeration of the concepts in the problem space. Here, the team is trying to get an overview of what they have done so far. This is a form of divergent thinking, jumping between concepts in a seemingly defocused fashion: *"Should we model any more parts? I mean there needs to be some kind of simulation engine [..] And we need probably something like some statistic engine [..] And UI of course."*.

An additional style exhibited by DT3 is the presence of creative events that map problem space to solution space in a convergent manner. That is, the team jumps between requirements (problem space) and UI or implementation details (solution space) to understand which concepts from the problem space already exist in the solution space: *"Okay. What did you think of, of. What did you envision [..]" "it is [sic] enough to just have some a kind of a grid of with varying lengths between the intersections. What could that be? [..]" "We still need the item pane at the bottom for having traffic light or these sensor thingies or whatever it was in the requirements."*.

5 | DISCUSSION

In this section we discuss the results in relation to existing work. We discuss results of RQ1 in Section 5.1, RQ2 in Section 5.2, and RQ3 in Section 5.3.

5.1 | RQ1 communication gaps caused by social and geographic distances affect design thinking

Our results provide a potential explanation as to how design communication is hampered in geographically distributed teams, namely, by reducing problem space exploration in favor of a more solution-oriented communication. Clark et al[65] state that in a collaborative setting, individuals keep on discussing and sharing knowledge until they reach a mutual understanding of the discussed argument. In problem space exploration, developers extensively exchange and complement their knowledge of the domain in order to reach a shared understanding of the problem space.[66] Achieving a shared understanding of the problem space might therefore be hampered in geographically distributed teams. This is an interesting addition to the more general observation that geographic and social distances can hamper communication and coordination.[8,41,42]

In summary, we observe that geographic distribution reduces problem space exploration in collaborative software design tasks compared with the colocated setting, thus answering RQ1. We believe that this finding explains in more detail how communication is affected by geographic and social distances.

5.2 | RQ2 technology and social factors, the common challenges to designing in a distributed way

We have observed that gestures or facial expressions were hard or impossible to recognize. This was the case despite the provided videoconferencing facilities, for example, when they were moving in front of the interactive whiteboard. This is a technological challenge known from related work on distributed work.[37]

Lack of common knowledge, challenge referring to individuals not knowing whether they have the same level of knowledge or understanding of the subject matter as their remote counterpart[8] was encountered. In the Gap model,[42] this challenge relates to cognitive distance. This issue could be reduced by collaborating with developers from the same department or school of thought.

The results of the poststudy questionnaire confirm Proposition B (tool support and social factors are the most frequent challenges in GSE). However, sociocultural factors might be underrepresented, since participants are likely more aware of technical limitations, for example, due to cognitive bias.

It is interesting to note that only one participant noted cultural differences as a perceived challenge. In Case 2, German developers were partnered with Swedish developers. As the German and Swedish cultures do have large differences on the Hofstede culture dimensions,[44] especially in masculinity and uncertainty avoidance, we would have expected a more noticeable influence of culture. However, as noted above, participants might not have perceived this challenge strongly.

In addition to sociotechnical factors, the Gap model[42] includes artifact-related distances, for example, semantic distances between multiple specifications. In our study, these distances were excluded by design and therefore do not show up in the perceived challenges. However, even if they were included, we believe that they would have a similar influence in both colocated and distributed settings.

In summary, we can answer RQ2 stating that both technical and social challenges are commonly encountered in collaborative, distributed software design.

While we cannot directly infer a causal connection between our results for RQ1 and RQ2, it is likely that the observed challenges for RQ2 play a major role in how *design thinking* is affected in Case 2. To be able to efficiently explore problem and solution space and to align them iteratively, it is essential to be aware of the current situation. However, the top two challenges, a lack of awareness of the remote counterpart and a lack of common understanding, indicate that this is not sufficiently possible. This can lead to less overall communication or lead to more explicit communication, for example, by not exploring the problem space sufficiently (as noted in **Observation 1**). The observed reduction in problem space exploration could be explained in two ways. First, individuals typically rely on previous knowledge for problem solving instead of a thorough analysis.[23] This effect might be amplified by sociotechnical barriers. Second, due to different barriers introduced by geographic distance, developers in Case 2 might do more tacit, internal *design thinking* and only present ideas after a number of internal iterations through the *design thinking* loop. This could lead to a lower overall quality of the resulting product, since not all ideas or solution candidates are discussed by all developers involved in the collaboration. For example, the distributed developers did not have a direct face-to-face communication and, therefore, missed eventual facial expressions and hand gestures, which often give a hint whether or not an argument is mutually understood.

Overall, we formulate the causal relation between our findings for RQ1 and RQ2 as a hypothesis for future work:

Research Hypothesis 1 (RH 1).

Lack of awareness of the remote counterpart and lack of common understanding reduce the amount of problem space exploration in distributed design thinking.

5.3 | RQ3 communication gaps caused by social and geographic distances do not clearly affect design creativity

According to Dorst and Cross,[4] creative events are "highly emotional steps," where subjects feel they have understood or uncovered "the underlying problem" behind or "the difficulty" with a task. Dorst and Cross[4] observed such events in all their study participants. In contrast, we did not observe events that clearly followed this pattern in any of the teams. Therefore, we relaxed the coding guidelines towards a more general notion of insights or realizations regarding the problem or the solution, as discussed in Section 3.

Even with the relaxed coding guidelines, we notice an overall low amount of divergent discussions leading to creative events, compared with the amount of convergent discussions that result in realizations. As noted by Goldschmidt[64] and Gabora,[30] a combination of divergent and convergent discussions is believed to increase design creativity. Hence, this indicates that our teams exhibit relatively little design creativity and that encouraging more divergent discussions could increase their creativity. We formulate this potential improvement as another hypothesis for future work:

Research Hypothesis 2 (RH 2).

Encouraging more divergent discussions in software design tasks increases design creativity.

There are several potential arguments why we did not observe creative events as described by Dorst and Cross and why divergent discussions were comparably few. First, our study design might have been too vague, broad, or permissive so that true insights were restricted. For instance, given the broad scope of the task, participants might have generally preferred a shallow exploration of the problem, covering a large proportion of the requirements in favor of a deeper discussion. Secondly, Dorst and Cross[4] study industrial design, as opposed to this SE task. That is, the domain and the subject of software design could inhibit creative events as described by Dorst and Cross. One argument for this option is that software design is on a high level of abstraction, trying to avoid detailed implementation decisions. In the given task, this level of abstraction could make it possible that participants stop the discussions before reaching the complexity necessary to truly understand the underlying difficulties. Similarly, the multiple different viewpoints from which a software design task can be addressed in contrast to a purely physical device could make the software design process different from industrial design. Thirdly, in our setup, pairs of participants are discussing a design task, without the possibility to ask intermediate questions. In contrast, Dorst and Cross[4] study a setup in which individual participants solve a design task in which they are able to interact with an *expert*. In several of our teams, for example, DT1 and DT3, we noticed that participants made assumptions or decided to defer the issue at points in the discussion where the complexity of the problem became apparent. For instance, CT2 had several statements similar to the following excerpt, after which they would make an assumption or continue with another topic: *"Yeah, I think we're going to have to rely on professor E for creating the details about the theory of how that works."*. Having the option to clarify would have either solved the issue, or given the team enough information to continue discussions, potentially leading to "true" creative events. Finally, as indicated by Mohanani et al,[67] framing desired as requirements can inhibit design creativity, something that indeed could have been the case in our task.

Regarding the connection between design creativity and design thinking, we observe that the teams that quickly jump between topics and make many assumptions seem to be the teams that exhibit most problem space exploration. In the colocated case, CT2 has with 43.8% by far the highest proportion of problem space exploration, followed by CT1 with only 29.4%. Investigating the creative events, we notice that this team exhibits a clear pattern of making assumptions whenever they come across a new issue or complexity that they cannot solve immediately: *"So this can get fairly complex very fast because you have all these intersections. [..] Lets assume this, lets assume it's inside a city so…[..] We'll talk to Professor E about it.]"*. This allows the team to quickly cover a large amount of concepts from the problem space, potentially leading to the high proportion of problem space exploration. In the distributed case, DT1 has 32.1% problem space exploration, followed by DT3 with 17.2%. As for CT2, DT1 makes a large number of assumptions, without going in depth with the respective issue. In the analysis of creative events, this is reflected by a majority of events that relate to the solution space. All events that purely relate to the problem space are events in which a participant reads a requirement and realizes that a concept/a requirement is missing in the solution, for example, *"Here it says: Students must… with or without the option to have sensors. So it is stated that it is related to the intersection, not to the road*. There is no evident correlation between the number of creative events and the proportions of problem and solution space exploration. Similarly, we do not see any correlation between the number of convergent/divergent discussions leading to a creative event. This indicates that creative events cover a dimension orthogonal to the design thinking phenomenon.

A final observation worth discussing is the notion of strategies how to reach creative events. Dorst and Cross[4] state that designers seem to have a strategy to reach creative events: "They search through the information by asking a quasi-standard set of questions, such as "capability of the company" and "available investment." Apparently, they have a set of expectations about the answers to these questions. [..] In doing this, they check the information related to the assignment to build up a general image and to look for surprises."[4] In our teams, we observe a number of similar strategies. First, enumeration of problem/solution space concepts is used by most teams at the beginning and the end of their sessions to find

complex or missing concepts. While the teams start the task with an expectation of what is needed, this enumeration helps them to find any surprising concepts that they did not initially consider or that are complex to translate to the solution space. At the end of their sessions, the enumeration of problem space concepts is used as a form of reconciliation, to identify whether any obvious concepts or requirements were missed or not addressed in the solution space. Finally, we notice several teams taking into account real-life situations when they are stuck in the solution space. That is, if there is an issue they cannot directly solve, they try to understand how this works in reality, in the hope that this will help them solve their problem.

6 | THREATS TO VALIDITY

In the following, we will describe potential threats to validity and how we addressed them in our study. We follow the classification into different threats according to Maxwell,[68] as they fit a broad spectrum of different scientific world views.[69] For an account how this classification compares to more common ways of discussion validity in SE, such as internal, external, conclusion, and construct validity, please refer to Petersen and Gencel.[69]

6.1 | Descriptive validity

Descriptive validity in qualitative research is the "factual accuracy" of the reported data.[68] It is not concerned with any interpretation of the data, only whether it is accurately represented in the account.

In our study, we used clean verbatim transcripts of the recorded video sessions. As such, accounts of data are not relying on memory. However, there is a minor threat to descriptive validity since we used transcripts for data analysis instead of the video recordings. This could have affected our analysis, since the tone of the voice, gestures, or facial expressions might have contained important information on emotional state or emphasis.

6.2 | Interpretive validity

Interpretive validity concerns whether "conclusions/inferences drawn [are] reasonable given the data representing an objective/subjective truth."[69] That is, in contrast to descriptive validity, the interpretation of data is a central concern.

To reduce threats to interpretive validity, we employed a number of measures. First, we relied on two established theoretical concepts: the design thinking concept[3] and that of convergent/divergent thinking.[64] For coding the data, we used written guidelines such as the coding guidelines presented in Appendix B. Multiple researchers then applied these guidelines to the same data and discussed differences. That is, three authors divided the transcripts among each other and applied the guidelines, while one author with approximately 8 years of experience in qualitative analysis coded parts of all transcripts and discussed with the respective author responsible for that transcript. In cases where it proved difficult to exactly apply the guidelines, we relaxed them. For example, this was the case for creative event coding, as emotions and emphasis were not clearly identifiable in the written transcripts.

Furthermore, we described the case study design and process in detail in Section 3, to allow readers to assess the validity in more depth. Finally, we use the published task description by Petre and Van der Hoek,[13] whose material can be obtained online (https://www.ics.uci.edu/design-workshop/videos.html). This enables researchers to replicate our study. To the extent this is feasible in qualitative case studies, this should enable a reproduction of our results under comparable contexts.

6.3 | Theoretical validity

Theoretical validity refers to the validity of the theoretical concepts employed in a study and the relationships between them.[68]

Design thinking is a concrete phenomenon, a natural and ubiquitous human activity during problem-solving processes.[70] We assessed **explicit** *design thinking*, that is, *design thinking* expressed verbally. However, *design thinking* is a cognitive activity and can happen also implicitly inside the mind of the thinking developer (tacit *design thinking*). This is a threat to theoretical validity, as the *design thinking* processes captured in our study might not represent the overall *design thinking* activity.

We collected data on challenges to distributed *design thinking* by asking the distributed developers to report their perceived challenges during the design sessions. These challenges are subjective and may vary from one developer to another. Furthermore, developers might have forgotten

to report some of the challenges that they experienced during the design session. To mitigate this issue, we additionally tried to triangulate the reported challenges with those observed during the distributed design sessions.

The heterogeneity and selection of subjects is a further threat to theoretical validity. While all teams consisted of members with professional experience, this experience differed. Specifically, the colocated teams had on average 19 years of professional experience, the distributed teams on average 8 years. The distributed teams were recruited among personal contacts of the authors, while we did not have control over the recruitment in Case 1. This might have affected how the teams interacted and how they solved the task. While a more homogeneous sample would have been beneficial, recruiting professionals for SE studies is a challenge. Furthermore, since this study is a case study, the contextual factors do in any case prevent us from making precise or generalizable statements regarding the study outcomes.

As a last threat to theoretical validity, the theoretical constructs we used in our study might not have been defined well enough to ensure similar understanding among all researchers. To reduce this threat, we used written coding guidelines as discussed for interpretive validity. The coding schema used for the analysis of design thinking was furthermore based on an existing schema.[61] Furthermore, we regularly discussed intermediate coding results.

6.4 | Internal generalizability

Internal generalizability is the extent to which generalization "within the community, group, or institution studied to persons, events, and settings that were not directly observed or interviewed"[68] is possible.

The level of expertise and experience in software architecture design might influence the *design thinking*. Novice and expert designers are observed to have different strategies to solve ill-defined problems.[66,71] Experts are able to store and retrieve information in larger cognitive chunks than novices. Moreover, experts concentrate on recognizing underlying principles rather than focusing on the surface feature of problems. The results of this study expose and explain the *design thinking* of expert developers. To understand the behavior of novice developers, we call for replication.

Different developers might perceive different challenges to their GSE experience. The reported GSE challenges by the participants of Case 2 are in line with the challenges reported in GSE literature and practice. However, we cannot guarantee that other developers would report the same challenges.

The participants in Case 2 used an interactive whiteboard (i.e., OctoUML) for their distributed collaboration. This could have affected our comparison of the *design thinking*. Moreover, using new tools often causes a learning effort. To mitigate these two issues, we customized OctoUML to resemble a regular whiteboard as much as possible. Furthermore, we deployed OctoUML on an interactive whiteboard and made use of Skype for live communication. This allowed the developers to collaboratively communicate and concurrently sketch and at the same time. In addition, all distributed developers had a short hands-on experience (i.e., 2–3 min) to test the environment by collaboratively sketching on the shared canvas of OctoUML. As discussed in Section 3, we measured the usability of OctoUML through a standard SUS questionnaire.[58] The average SUS score of 74.17 ± 5.63 (good usability according to[60]) indicates that OctoUML has a reasonable usability and does not perform significantly worse than a state-of-the-art tool for distributed collaboration. Hence, the use of OctoUML should not present a confounding factor in our study. However, we also observed several discussion episodes in Case 2 that were interrupted by tool and connection problems. This might have affected the design creativity of the teams, as they were interrupted in their thinking.

The design sessions lasted about 90 min, that is, some teams finished the task early even though they had more time. Thus, the participants might have suffered from fatigue that could have led to less *design thinking* or design creativity. We assume that fatigue would have a similar effect in both cases and therefore not affect our comparison.

6.5 | External generalizability

External generalizability is the extent to which generalization "to other communities, groups, or institutions"[68] is permitted by the study.

By design, case studies have a very limited external generalizability stemming from the fact that a topic is studied within its context and that there is little to no control. Therefore, we cannot claim that our findings are generalizable, that is, different projects in different domains might have different results. However, we described the case context as detailed as possible in order to allow practitioners to decide whether or not the findings might apply in their own case context.

We asked participants to perform a specific software architecture design task. In industrial settings, design tasks can vary substantially, for example, in terms of size, terminology, language, and level of detail. In particular, our assignment is a green-field case, that is, involving both the analysis of the problem domain and the synthesis of a solution from scratch. This also limits the external generalizability of the findings. Moreover, we underline that our study involved expert software architecture designers. Different results might be obtained if novice designers were

involved in this study. Notwithstanding the aforementioned threats, many of our findings are consistent with the findings of other research on global software development.

7 | CONCLUSION

In this paper, we reported a multiple-case study exploring the effect of communication gaps caused by social and geographic distances on design thinking (RQ1) and design creativity (RQ3) and the challenges perceived by geographically distributed teams (RQ2). We observed that distributed developers did less *design thinking* than the colocated developers, specifically in problem space exploration and in the alignment between the problem and solution space (**Observation 1**). Participants in the distributed case perceived a lack of awareness of the remote counterpart and a lack of common understanding as the main challenges in distributed design. We hypothesize (**RH 1**) that these perceived challenges might be causing the observed reduction in *design thinking*. However, we did not observe obvious differences in design creativity between colocated and distributed teams (**Observation 3**), apart from connection issues that caused interruptions in the distributed teams. Finally, we observe almost no creative events that strictly follow the description by Dorst and Cross[4] and only few divergent discussions leading to creative events. We hypothesize (**RH 2**) that guiding teams to increase the amount of divergent discussions could increase design creativity and therefore the amount of creative events.

7.1 | Implications for research

Our study extends the body of knowledge in GSE, since it offers an explanation how geographic and social distances affect communication, namely, by reducing problem space exploration. While it is known that GSE might hamper communication, our findings concretize this knowledge. This is especially valuable since our analysis of *design thinking* does not rely on subjective perceptions and can therefore complement data based on participant perceptions. Similar analyses could be used in future work to measure the effect of potential solutions for designing in distributed settings. To this end, we developed a general coding schema for analyzing design discourses, based on existing work by Weinreich et al..[61]

Our findings for RQ2 mainly confirm existing work in GSE. Whether or not there is a causal connection between our findings for RQ1 and RQ2 is as of now hypothetical (**RH 1**). Specifically, our findings rely mainly on the perceptions of study participants. Since social challenges in particular might be implicit and therefore not visible in a self-evaluation, we see the need for studying this connection in more detail, that is, how sociotechnical barriers, such as culture or beliefs, affect *design thinking* in distributed teams. Increasing the understanding of these factors would contribute to a more efficient and effective remote collaboration and thus result in products of higher quality.

Our findings for RQ3 offer mainly methodological insights for SE research. In contrast to *design thinking*, we do not observe any clear differences in design creativity between colocated and distributed teams and no apparent correlation between the creative events and the observed *design thinking*. Factors that could play a role here are differences between SE and industrial design, which is the context in which they were described by Dorst and Cross,[4] the abstract and multiview nature of software design in contrast to purely physical devices, or methodological shortcomings of our study. However, it could indeed be the case that problem space exploration is reduced in distributed teams but design creativity is not. Hence, the distributed teams could simply be more effective in their problem space exploration. To our knowledge, there exists no previous work in SE that analyses design creativity by identifying creative events and only little work that focuses on design creativity, for example, Mohanani et al.[67] Therefore, similar studies are needed to evaluate the usefulness of this analysis approach.

Another interesting observation for RQ3 is the relative lack of divergent discussions that lead to creative events in our teams. We hypothesized in previous work, which is extended by this paper, that future work could build on the reminder cards proposed by Tang et al,[33] to suggest a set of reminder cards that foster problem space exploration. Our findings suggest that encouraging more divergent discussions should be another focus of such a new approach, as it has the potential to increase overall design creativity in both colocated and distributed teams (**RH 2**). The strategies for reaching creative events mentioned by Dorst and Cross[4] could be another addition to such an approach.

7.2 | Implications to practice

While practitioners might be aware of negative effects of geographic distribution, our findings can help them understand how this manifests in practice. This knowledge can in turn help them to decide how to engage in GSE. A potential decision might be to limit tasks that require substantial problem space exploration to colocated teams, for example, when features or products with large uncertainty are designed. Similarly, in distributed design sessions, practitioners might decide to counteract the technological challenges observed in this study, for example, by adding video conferencing that shows the faces and gestures of the involved designers. These decisions could positively affect development achievement and, therefore, product quality and customer satisfaction.

In addition to technical factors, cultural and other social barriers might increase due to distribution. Our findings confirm existing works with respect to the importance of these nontechnical factors and can reinforce practitioners in initiatives that aim to mitigate these barriers.

7.3 | Future work

In addition to the several directions for future work outlined in this paper, we plan to extend our study to further SE activities. In particular, we plan to analyze whether problem space exploration is reduced in a similar fashion in software modeling and requirements engineering activities. In addition, we want to further investigate the sociocultural dimension in this context, for example, by controlling for different cross-local practices or knowledge gaps between different geographic sites. Finally, our findings for RQ3 questions the extent to which the *design thinking* and design creativity analyses are useful in SE. We believe that further studies in this direction need to be conducted, in order to clearly evaluate the potential of these theories in SE.

ACKNOWLEDGMENTS

We would like to acknowledge the subjects participating in our study for their valuable time. Furthermore, we thank André van der Hoek for sharing the dataset for the colocated teams with us.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

AUTHOR CONTRIBUTIONS

All the authors contributed in writing and reviewing this manuscript. The design of the "distribution" case study was performed by Rodi Jolak and Michel R.V. Chaudron. The "distribution" case study was conducted by Rodi Jolak and Andreas Wortmann. The two case studies were analyzed and discussed by Rodi Jolak, Andreas Wortmaan, Grischa Liebel, and Eric Umuhoza.

FINANCIAL DISCLOSURE

Not available.

DATA AVAILABILITY STATEMENT

The research data and transcriptions related to this study are available and can be shared with interested people. Please send a request to rodi.jolak@cse.gu.se.

ORCID

Rodi Jolak https://orcid.org/0000-0001-5656-9253
Andreas Wortmann https://orcid.org/0000-0003-3534-253X
Grischa Liebel https://orcid.org/0000-0002-3884-815X
Eric Umuhoza https://orcid.org/0000-0002-2451-8897
Michel R. V. Chaudron https://orcid.org/0000-0001-7517-6666

REFERENCES

1. Jolak R, Umuhoza E, Ho-Quang T, Chaudron MRV & Brambilla M Dissecting design effort and drawing effort in UML modeling. In: IEEE; 2017: 384–391.
2. Petre M, Van der Hoek A, Quach Y. *Software Design Decoded: 66 Ways Experts Think*. MIT Press; 2016.
3. Cross N. *Design Thinking: Understanding How Designers Think and Work*. Ther Ber; 2011.
4. Dorst K, Cross N. Creativity in the design process: Co-evolution of problem–solution. *Des Stud*. 2001;22(5):425-437.
5. Ebert C, Kuhrmann M, Prikladnicki R. Global software engineering: Evolution and trends. In: IEEE; 2016: 144–153.
6. Damian D, Moitra D. Guest editors' introduction: Global software development: How far have we come? *IEEE Softw*. 2006;23(5):17-19.
7. Paasivaara M, Lassenius C. Collaboration practices in global inter-organizational software development projects. *Softw Process: Improv Pract*. 2003; 8(4):183-199.
8. Herbsleb JD. Global software engineering: The future of socio-technical coordination. In: IEEE; 2007: 188–198.
9. Olson JS, Olson GM. Working together apart: Collaboration over the internet. *Synth Lect Hum-Cent Inf*. 2013;6(5):1-151.
10. Herbsleb JD, Moitra D. Global software development. *IEEE Softw*. 2001;18(2):16-20.
11. Kan JWT, Gero J. Studing software design cognition. In: Hoek AVD, Petre M, eds. *Software Designers in Action: A Human-Centric Look at Design Work*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. Chapman and Hall/CRC; 2013.
12. Williams C, Storey MA, Ernst NA, Zagalsky A, Kalliamvakou E. Methodology Matters: How we Study Socio-Technical Aspects in Software Engineering *arXiv Preprint*: 1905; 12841 2019.

13. Petre M, Van der Hoek A. *Software Designers in Action: A Human-Centric Look at Design Work*. Chapman and Hall/CRC; 2013.
14. Jolak R, Wortmann A, Liebel G, Umuhoza E, Chaudron MRV. The design thinking of co-located vs. distributed software developers: Distance strikes again!. In: IEEE/ACM.; 2020: 106–116.
15. Kimbell L. Rethinking design thinking: Part I. *Des Cult*. 2011;3(3):285-306.
16. Buchanan R. Wicked problems in design thinking. *Des Issues*. 1992;8(2):5-21.
17. Martin R. *The Design of Business: Why Design Thinking Is the Next Competitive Advantage*. Cambridge, MA; 2009.
18. Dobrigkeit F, Paula DD. Design thinking in practice: Understanding manifestations of design thinking in software engineering. In: ACM; 2019: 1059–1069.
19. Petre M, van der Hoek A, Bowers DS. *Software Design as Multiple Contrasting Dialogues*. Psychology of Programming Interest Group; 2019.
20. Lindberg T, Meinel C, Wagner R. *Design Thinking: A Fruitful Concept for IT Development?* Springer; 2011:3-18.
21. Brooks FP Jr. *The Design of Design: Essays From a Computer Scientist*. Pearson Education; 2010.
22. Jolak R, Wortmann A, Chaudron MRV, Rumpe B. Does distance still matter? Revisiting collaborative distributed software design. *IEEE Softw*. 2018; 35(6):40-47.
23. Christiaans H, Almendra RA. Accessing decision-making in software design. *Des Stud*. 2010;31(6):641-662.
24. Baker A, van der Hoek A. Ideas, subjects, and cycles as lenses for understanding the software design process. *Des Stud*. 2010;31(6):590-613.
25. Razavian M, Tang A, Capilla R, Lago P. In two minds: How reflections influence software design thinking. *J Softw: Evol Process*. 2016;28(6):394-426.
26. Dorst K. The core of 'design thinking' and its application. *Des Stud*. 2011;32(6):521-532.
27. Kruger C, Cross N. Solution driven versus problem driven design: Strategies and outcomes. *Des Stud*. 2006;27(5):527-548.
28. Wiltschnig S, Christensen BT, Ball LJ. Collaborative problem–solution co-evolution in creative design. *Des Stud*. 2013;34(5):515-542.
29. Ahmed S, Christensen BT. An in situ study of analogical reasoning in novice and experienced design engineers. *J Mech Des*. 2009;131(11).
30. Gabora L. Revenge of the "neurds": Characterizing creative thought in terms of the structure and dynamics of memory. *Creat Res J*. 2010;22(1):1-13.
31. Maiden N, Robertson S. Integrating creativity into requirements processes: Experiences with an air traffic management system. In: IEEE; 2005: 105–114.
32. Boden MA. *The Creative Mind: Myths and Mechanisms*. Psychology Press; 2004.
33. Mohanani R, Turhan B, Ralph P. Requirements framing affects design creativity. *IEEE Trans Softw Eng*. 2019.
34. Nguyen L, Shanks G. A framework for understanding creativity in requirements engineering. *Inf Softw Technol*. 2009;51(3):655-662.
35. Ó Conchúir E, Holmström Olsson H, Ågerfalk PJ, Fitzgerald B. Benefits of global software development: Exploring the unexplored. *Softw Process: Improve Pract*. 2009;14(4):201-212.
36. Khan AA, Keung J, Niazi M, Hussain S, Shameem M. GSEPIM: A roadmap for software process assessment and improvement in the domain of global software development. *J Softw: Evol Process*. 2019;31(1):e1988.
37. Dourish P, Bellotti V. Awareness and Coordination in Shared Workspaces. In: CSCW'92. ACM. ; 1992: 107-114
38. Gutwin C, Greenberg S, Roseman M. *Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation*. Springer; 1996: 281-298.
39. Clark HH. *Using Language*. Cambridge University Press; 1996.
40. Monk A. Common ground in electronically mediated communication: Clark's theory of language use. In: *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*; 2003:265-289.
41. Damian D, Marczak S, Kwan I. Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. In: IEEE; 2007: 59–68.
42. Bjarnason E, Smolander K, Engström E, Runeson P. A theory of distances in software engineering. *Inf Softw Technol*. 2016;70:204-219.
43. Lehmann-Willenbrock N, Allen JA, Meinecke AL. Observing culture: Differences in US-American and German team meeting behaviors. *Group Process Intergroup Relat*. 2014;17(2):252-271.
44. Hofstede G. *Culture's Consequences: Comparing Values, Behaviors, Institutions and Organizations Across Nations*. Sage Publications; 2001.
45. Shah H, Nersessian NJ. Cultural Models and their Interplay in Global Software-Engineering Practice. In: IEEE; 2015: 13-22.
46. Matthiesen S, Bjørn P, Trillingsgaard C. Attending to implicit bias as a way to move beyond negative stereotyping in GSE. In: ACM; 2020: 22-32.
47. Bjørn P, Esbensen M, Jensen RE, Matthiesen S. Does distance still matter? Revisiting the CSCW fundamentals on distributed collaboration. *ACM Trans Comput-Hum Inter (TOCHI)*. 2014;21(5):1-26.
48. Jolak R, Liebel G. Position Paper: Knowledge Sharing and Distances in Collaborative Modeling. In: IEEE; 2019: 415-416.
49. Cherubini M, Venolia G, DeLine R, Ko AJ. Let's go to the whiteboard: How and why software developers use drawings. In: ACM SIGCHI; 2007: 557–566.
50. Dekel U. Supporting distributed software design meetings: What can we learn from co-located meetings? *ACM SIGSOFT Softw Eng Notes*. 2005;30(4): 1-7.
51. Wu J, Graham TN. *The Software Design Board: A Tool Supporting Workstyle Transitions in Collaborative Software Design*. Springer; 2004:363-382.
52. Mangano N, Baker A, Dempsey M, Navarro E, van der Hoek A. Software design sketching with calico. In: IEEE/ACM; 2010: 23-32.
53. Hammond T, Gajos K, Davis R, Shrobe H. An agent-based system for capturing and indexing software design meetings. In: Gero J, Brazier F, eds. *Agents in Design*. Australia: Key Centre of Design Computing and Cognition, University of Sydney; 2002.
54. Jolak R, Vesin B, Chaudron MRV. OctoUML: An environment for exploratory and collaborative software design. In: 17. IEEE/ACM. ; 2017: 7-10.
55. Yin RK. *Case Study Research and Applications: Design and Methods*. Sage Publications; 2017.
56. Runeson P, Höst M, Rainer A, Regnell B. *Case Study Research in Software Engineering*. Wiley Online Library; 2012.
57. Olson GM, Olson JS. Distance matters. *Hum–Comput Inter*. 2000;15(2-3):139-178.
58. Brooke J. SUS: A retrospective. *J Usabil Stud*. 2013;8(2):29-40.
59. Tullis TS, Stetson JN. A comparison of questionnaires for assessing website usability. In: 1. Minneapolis, USA.; 2004.
60. Sauro J. *A practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC Denver, CO; 2011.
61. Weinreich R, Groher I, Miesbauer C. An expert survey on kinds, influence factors and documentation of design decisions in practice. *Future Gener Comput Syst*. 2015;47:145-160.
62. Koo TK, Li MY. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *J Chiropr Med*. 2016;15(2):155-163.

63. Saldaña J. *The Coding Manual for Qualitative Researchers*. Sage; 2015.
64. Goldschmidt G. Linkographic evidence for concurrent divergent and convergent thinking in creative design. *Creat Res J*. 2016;28(2):115-122.
65. Clark HH, Brennan SE. Grounding in communication. In: Resnick LB, Levine JM, Teasley SD, eds. *Perspectives on Socially Shared Cognition*. 13. American Psychological Association; 1991:127-149.
66. Cross N. Expertise in design: An overview. *Des Stud*. 2004;25(5):427-441.
67. Maxwell J. Understanding and validity in qualitative research. *Harvard Educ Rev*. 1992;62(3):279-301.
68. Petersen K, Gencel C. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: IEEE; 2013: 81-89.
69. Cross N. Designerly ways of knowing: Design discipline versus design science. *Des Issues*. 2001;17(3):49-55.
70. Ahmed S, Wallace KM, Blessing LT. Understanding the differences between how novice and experienced designers approach design tasks. *Res Eng Des*. 2003;14(1):1-11.
71. Tang A, Bex F, Schriek C, Van der Werf JMEM. Improving software design reasoning—A reminder card approach. *J Syst Softw*. 2018;144:22-40.

How to cite this article: Jolak R, Wortmann A, Liebel G, Umuhoza E, Chaudron MRV. Design thinking and creativity of colocated versus globally distributed software developers. *J Softw Evol Proc*. 2023;35(5):e2377. doi:10.1002/smr.2377

APPENDIX A: CODING OF DESIGN THINKING

Table A1 displays the codebook for the coding task in *Design Thinking*. We have taken actual quotes from the dataset to exemplify the codes. The only exception is the *Deployment Decision* code, which was never used in the actual analysis. Therefore, we use a fictional quote instead.

TABLE A1 Codebook for design thinking coding

DT space	Code	Description	Example
Problem space	Assumptions	Assumptions about domain and context.	"And then the timing scheme. I guess there needs to be a timer for."
	Definition of use cases	Definition of traffic flow use cases.	"I mean let's take your example here. Let me draw. So if we take this example and you just want to move this intersection over here instead."
	Architectural drivers	Architectural drivers such as security, usability, maintainability.	"Of course we can represent this here but we should also keep in mind that this could be changed in the future and that we should design the system that is able to change this."
	Identification of actors	Identification and definition of actors.	"Well, I mean so the end users seem to be the students, and the professor. Students need to be able to build these roads but also see the results of what happens when they make changes."
	Scoping	Decision on what is part of the system.	"[..] I just mark it blue to say we think it would be useful and should we go in this direction but to make a clear difference to what was our initial purpose regarding the requirements. Is that okay?"
Solution space	Behavioral Decision	Order of actions, initiative, synchronization, concurrency, etc.	"Doing the simulation, what kind of feedback are we getting?[..] So the cars keep appearing, going, appearing, going."
	Communication/ coordination	Communication or coordination between different parts of the system or actors.	"And the car needs to tell the intersection what it wants to do. So the car knows what it wants to do, where it's going and the intersection tells the car, yeah."
	Data abstraction	Choice of data abstraction (e.g., data structures, message content).	"Okay, so what basic data structures are we looking at here?We need something that represents intersections."
	Deployment Decision	Distribution of software artifacts, deployment targets.	"And then we could have the simulation running on the cloud in case we have to scale it or it does not meet our performance requirements."
	Design Principle	Design principles and guidelines, e.g., layering, architectural style.	"I'm thinking in terms of model-view-controller. Say start with a model and start with the simplest model to get the simplest intersection flowing with cars and timing."
	Functionality	Functionality of the system.	"And as you drag it out it kind of shows you how many cars…"
	Implementation Decision	Details of the implementation, e.g., algorithms, variable naming.	"So, yeah I guess at a very base level we kick everything off with main. Right, n = new network, c = new clock…"
	Notation Decision	Modeling notation, e.g., type of diagrams, low/high-level design.	"So maybe once we've done that we can think about UML diagrams for our class structures…"

TABLE A1 (Continued)

DT space	Code	Description	Example
	Structural Decision	Coupling/relations between components, layering, abstraction, etc.	"Okay, so there's a road, it has a name, [..]. And then where two roads cross, so that can cross many roads…"
	Technology Decision	Employed technology, e.g., platforms or libraries.	"Okay, and then there's the density controls. So I guess we could have some pre-fabricated for them."
	Usability/UI	Usability and user interface decisions.	"[..] let's say we have here a big picture and I guess in the middle we would have something like the edit area, where everything is also shown."

APPENDIX B: CODING OF CREATIVE EVENTS

Dorst and Cross[4] discuss that bridges between the problem and solution space are built by identifying "key concepts." The authors write that "creative design involves a period of exploration in which problem and solution spaces are evolving and are unstable until (temporarily) fixed by an emergent bridge which identifies a problem–solution pairing. A creative event occurs as the moment of insight at which a problem–solution pair is framed."[4]

We will aim to identify these key events (called *Creative Events* in the following) by manually going through the textual transcripts. The identification is a subjective process that cannot be completely formalized. For instance, keywords have only little value as they are depending on communication style and language. The following points are indicators that a Creative Event occurs:

1. Subjects understand a connection that they were unawares beforehand. This simplifies an issue they were dealing with so far, resulting in a solution to a part of the assignment/task.
2. In contrast to (1), subjects identify an issue they were unawares before, leading to a complication of the overall task.
3. Subjects find an appealingly simple solution to an issue. This greatly simplifies the issue or the overall task.
4. In terms of keywords, the subjects show sudden realization (e.g., "Aha," "Now I understand it," "Wait a moment, we haven't thought about…"). However, it is important to keep in mind that this is context dependent and can be hard to identify in the written transcriptions (e.g., a matter of fact "Now I understand what you are saying" vs. an emotional "Now I finally understood the assignment"). Dorst and Cross[4] state that Creative Events are generally "highly emotional steps."
5. Regularly, the subjects feel that they have understood or uncovered "the underlying problem" behind or "the difficulty" with the task.

After all Creative Events have been identified in all transcripts, we will look at the area around (±5 min) each Creative Event and try to characterize it. That is, we try to answer "What kind of discussion lead to the Creative Event?" and "What implications does the Creative Event have?". To do so, we will use descriptive coding, that is, we use codes similar to "hash tags" that describe what the subjects are discussing. We will not use any predefined (a priori) codes but rather use an open coding approach. Similarly, we do not define the stanza (the unit which is coded) but leave it up to the coder to decide which parts to assign a code to. This can also vary (e.g., one remarkable sentence gets a single code, while in another case only an entire paragraph is coded, and in yet another case, an entire paragraph does not get any code since it does not appear to be relevant for answering the questions).