# The Design Thinking of Co-located vs. Distributed Software Developers: Distance Strikes Again!

Rodi Jolak
rodi.jolak@cse.gu.se
Chalmers | Gothenburg University
Gothenburg, Sweden

Andreas Wortmann
wortmann@se-rwth.de
RWTH Aachen University
Aachen, Germany

Grischa Liebel
grischal@ru.is
Reykjavik University
Reykjavík, Iceland

Eric Umuhoza
eumuhoza@andrew.cmu.edu
Carnegie Mellon University-Africa
Kigali, Rwanda

Michel R.V. Chaudron
michel.chaudron@cse.gu.se
Chalmers | Gothenburg University
Gothenburg, Sweden

## ABSTRACT

*Context*: Designing software is an activity in which software developers *think* and make design decisions that ultimately shape the structure and behavior of software products. Currently, designing software is one of the least understood activities in which software developers engage. In a collaborative design setting, distances such as geographic, cultural, or social distance can lead to socio-technical challenges that potentially affect the way software is designed.
*Objective*: To contribute to an increased understanding of software design, we investigate how geographic distance affects collaborative software design.
*Method*: To this end, we conducted a multiple-case study exploring in depth the *design thinking* of co-located and distributed software developers in a collaborative design setting.
*Results*: We find that, compared to co-located developers, distributed developers practice less problem space exploration and focus instead more on the solution space. This could be related to different socio-technical challenges caused by distributed collaboration, such as lack of awareness and common understanding.
*Conclusion*: Our findings contribute to an increased understanding as to how software design is affected by geographic distance. Developers engaging in collaborative design need to be aware that problem space exploration is reduced in a distributed setting, which would adversely affect the development achievement and therefore customer satisfaction.

## CCS CONCEPTS

• **Software and its engineering → Designing software**; **Collaboration in software development**; **Abstraction, modeling and modularity**; *Software system models*; • **Social and professional topics → Geographic characteristics**; • **General and reference →** *Empirical studies*; Design.


*ICGSE '20, October 5–6, 2020, Seoul, Republic of Korea*

ACM ISBN 978-1-4503-7093-6/20/05…$15.00
https://doi.org/10.1145/3372787.3390438

## KEYWORDS

Software Engineering, Collaborative Design Thinking, Distance, Cognitive Aspects, Empirical Study, CSCW

**ACM Reference Format:**
Rodi Jolak, Andreas Wortmann, Grischa Liebel, Eric Umuhoza, and Michel R.V. Chaudron. 2020. The Design Thinking of Co-located vs. Distributed Software Developers: Distance Strikes Again!. In *15th IEEE/ACM International Conference on Global Software Engineering (ICGSE '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3372787.3390438

## 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 [26].

Developers *design* throughout the software engineering (SE) lifecycle [41], despite their different roles in a project. For example, developers do not only elicit requirements, they actually design the requirements by discussing and shaping these requirements with the contributing stakeholders. Developers also design their code by composing, analyzing, and evaluating algorithms, to ensure, e.g., an efficient algorithm run-time. Similarly, developers design use cases, interactions, user interfaces, and test cases.

To handle re-occurring *wicked* [43] and ill-structured problems during design, expert developers intuitively practice *design thinking* [13]. *Design thinking* 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 specially trained in *design thinking*.

As globally-distributed projects become the norm in SE [21] and lead to social, technical, and organizational challenges [15], it is likely that software design activities are affected as well. In particular, it is unclear to what extent the iterative cycle of *design thinking* is 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 [29]. This focus on technical aspects of design is problematic, as SE is a socio-technical endeavour, and further research controlling for human and social aspects is crucial for ensuring a successful engineering of software

systems [49]. In fact, Petre and Van Der Hoek [39] 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[1], investigating how *design thinking* takes place during software design in co-located and in distributed teams. Based on video data obtained by Petre and Van der Hoek [39] for the co-located case, and three distributed teams of professional software developers designing the architecture of a system at the whiteboard, we qualitatively analyze how software developers switch between the problem and solution space of *design thinking*, and how alignment takes place. 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?

Hence, the contributions of this paper are threefold:

(1) We analyze the *design thinking* process of software developers, leading to an increased understanding of software design activities.
(2) We qualitatively compare how *design thinking* differs between co-located 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.

In summary, we find that distributed teams practice less *design thinking* compared to co-located 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*.

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 discuss the results in Section 4 and the threats to validity in Section 5. We conclude and outline future work in Section 6.

## 2 RELATED WORK

In this section we discuss related work focusing on *Design Thinking* and *Global Software Engineering*.

### 2.1 Design Thinking

According to Kimbell [31], design thinking can be described as a cognitive style [13, 19], a general theory of design [7], or an organizational resource [36]. One school of thought considers *design thinking* as an activity that the subject is aware of, e.g., [17, 40]. In contrast, several authors understand *design thinking* as a theory that explains how subjects practise problem solving during a design task without necessarily being aware of it, e.g., [7, 13, 19]. In this study, we consider the latter understanding of *design thinking*.

Lindberg et al. [34] highlight that *design thinking* fosters three main activities (see Figure 1):

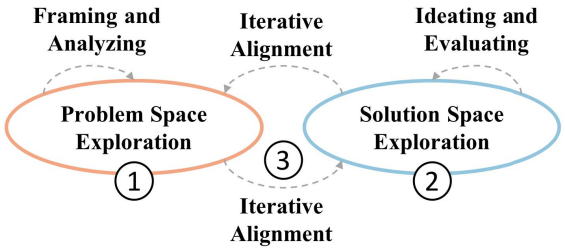


**Figure 1: Problem solving with design thinking [34]**

(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. [34] indicate that *design thinking* can broaden the problem understanding and problem solving capabilities in IT development processes. This is in line with Brooks [6], who considers *design thinking* an exciting new paradigm for dealing with problems in software and IT development.

Similar to Lindberg et al. [34], Dorst and Cross [19] 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.

In [40], Petre et al. 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 dialogues 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 dialogues provides a focus for design reasoning which helps to understand problems, manage complexity, and achieve enduring development success.

Dobrigkeit and de Paula [17] 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. [28] study how distributed software designers communicate and collaboratively make design decisions. They found that distance affects the quality of communication, and reduces the amount of design decisions that the developers take during their distributed collaboration. Our multiple-case study uses the same

[1] In this paper, we use the same data as in [28], but with different objective and analysis.

data used in [28]. In contrast, in this study we focus on analyzing *design thinking* and challenges to distributed collaboration.

Similarly, Christiaans and Almendra [9] 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 [2] 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*, i.e., by exploring the problem space, the solution space, and then repeatedly jumping back and forth to align the two spaces.

Razavian et al. [42] 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 [18].

In summary, *design thinking* is a cognitive style that fosters problem and solution space exploration and alignment between the two spaces. 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 Global Software Engineering

Global software engineering (GSE) is the practice of engineering software systems across geographical, socio-cultural, and temporal boundaries [23]. 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 [37]. However, these organizations often face numerous challenges, including poor quality of globally developed software [30].

Herbsleb [23] studies the impact of geographic distance on distributed collaboration. He suggests that co-location 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. [14], 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. [3] 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.
- 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. For instance, as part of a substantial body of work on culture, the Hofstede culture distances [24] are a well-known theory aiming to explain how different cultural backgrounds developed in families, schools, and organizations introduce differences in thinking and social actions. Geographic distribution, e.g., due to outsourcing, often lead to a more diverse mix of cultures which can give rise to communication gaps. For instance, Lehmann-Willenbrock et al. [33] observe that culture can introduce differences in the behaviour of distributed collaborating teams. In addition to the culture dimension, Bjørn et al. [4] 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 [25]. 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 [8]. 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 co-located case and interactive whiteboards in the distributed case) to better study the design thinking phenomenon.

Dekel [16] study co-located 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 [50], Calico [35], Metaglue [22], and OctoUML [27] (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* of co-located 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* phenomenon.

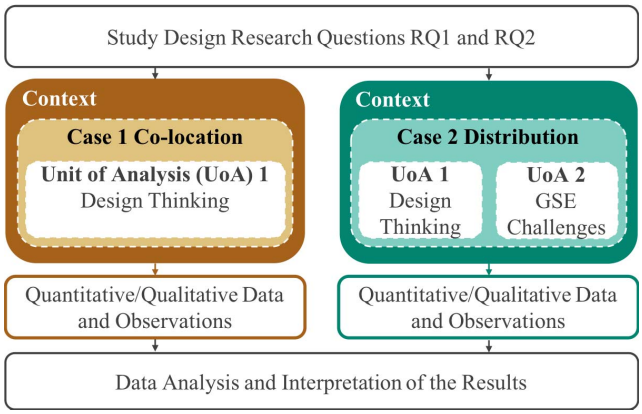


**Figure 2: Multiple-case study design featuring the two use cases of co-located and distributed software design**

Since it is hard to study *design thinking* 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 [51]. Our case study is an exploratory and inductive multiple-case study [44].

## 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** (*Co-location*): Collaborative co-located designing of software architecture using a whiteboard. This is a holistic case [51] with a single Unit of Analysis (UoA): *Design Thinking*. Here, we analyze the problem-solving cognitive style of developers.
- **Case 2** (*Distribution*): Collaborative distributed designing of software architecture using an interactive whiteboard. This is an embedded case [51] with two UoAs: *Design Thinking* 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 theoretical foundation consisting of theories, if possible, and propositions/hypotheses [44, 51]. Next, we detail the theoretical framework that we employ in this multiple-case study.

In this study, we make use of the *design thinking* phenomenon as a means to reason about how individuals solve problems. That is, we analyze their interactions with respect to the three *design thinking* activities described by Lindberg et al. [34]: 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*. The two cases are selected to predict possible contrasting results on *design thinking* by altering one condition: the geographic distance (co-location vs. distribution).

In addition to the *design thinking* phenomenon, we analyze our data with respect to Bjarnason's theory on distances in SE [3]. 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 two propositions, which then guide our case study design:

- **Proposition A**: We propose that geographic distance (co-location vs. distribution) does not affect *design thinking*. *If* the *design thinking* of the two cases varies considerably, *then* this indicates that geographic distance affects *design thinking*. This would encourage deeper investigations of the effect of distance on *design thinking*.
- **Proposition B**: We propose, based on the findings in [23, 38], 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.

## 3.3 Context

For both cases, we used the design challenge by [39], which can be consulted at https://www.ics.uci.edu/design-workshop. In this challenge, 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. 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.

Developers were given a printed copy of the design challenge and had a maximum of 2 hours 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 minutes to collect their thoughts and briefly explain the design.

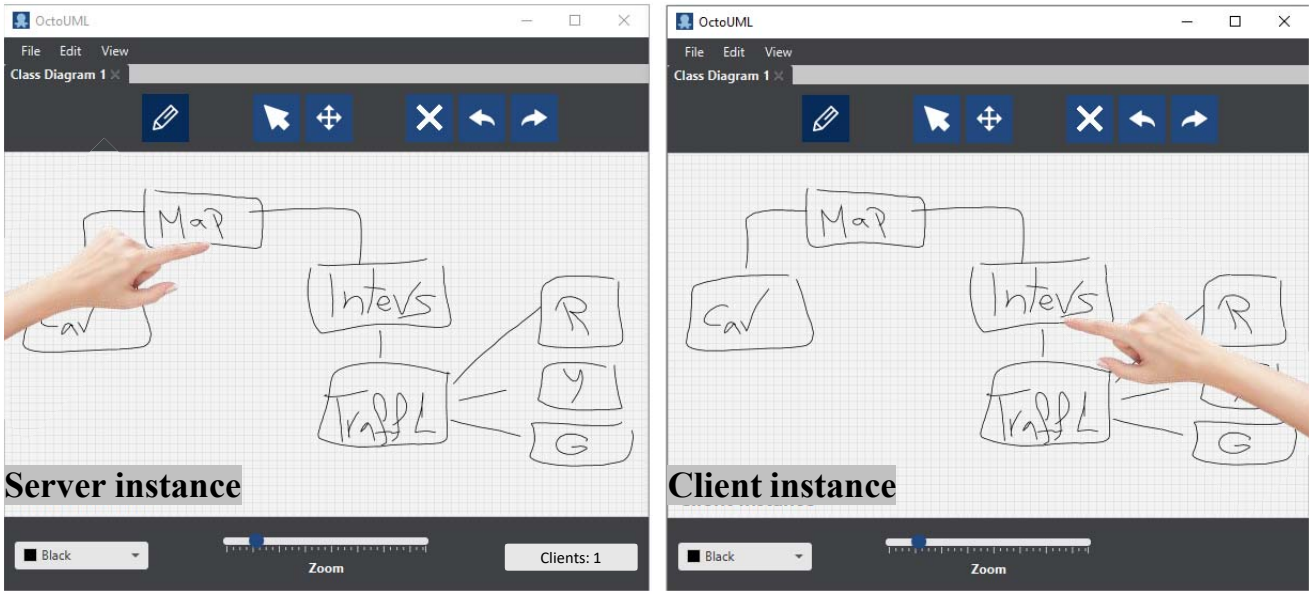


**Figure 3: The main views of the server and client instances of OctoUML**

For Case 1, we used the data set provided by [39], who performed the study with three (co-located) 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.

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 in our study had between three and seven years of professional software development experience in automotive and networking domains, and were selected based on convenience sampling.

Instead of a regular whiteboard, we used interactive whiteboards with a simplified version of OctoUML[2], an open source software design environment [27] 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 thinking) of the two cases. Similarly, choosing another commercial remote collaboration tool might have introduced other collaboration features that would have changed the experience compared to 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 the evaluate the usability of OctoUML. The reason is to understand to what 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 System Usability Scale is an easy, standard way of evaluating the usability of a system [5]. It is a form containing ten 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 [47]. Figure 4 shows the results of SUS evaluation. Overall, we observe that the perceptions regarding the usability of OctoUML were positive. Regarding the SUS score, OctoUML received an average SUS score of $74.17 \pm 5.63$, which can be interpreted as a grade of $B-$ (i.e., a good usability score) according to [45].

## 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 obtained from https://www.ics.uci.edu/design-workshop/videos.html).

[2]OctoUML Website: http://rodijolak.com/#octouml

[3]Skype Website: https://www.skype.com/en

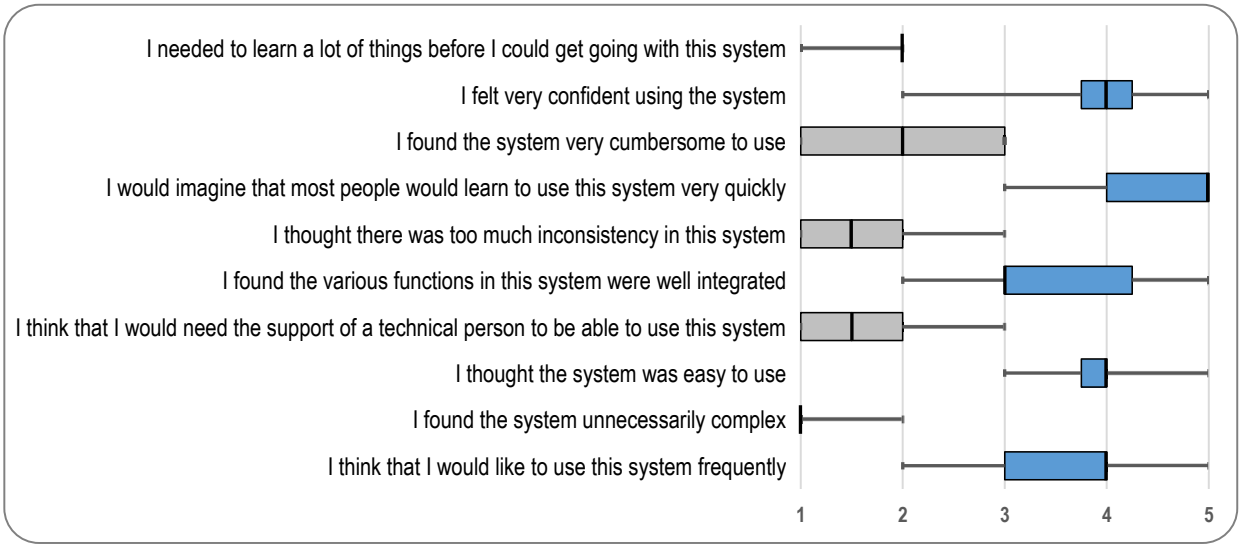


**Figure 4: The perceived usability of OctoUML**

*3.4.1 Design Thinking.* We analyzed the transcriptions of approximately 10 hours of design activity by six pairs of professional software developers, and performed a manual coding of more than 2000 conversation dialogues. 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 Rainer Weinreich et al. [48]. Dialogues in the transcriptions were then assigned codes (using NVivo[4]). If multiple codes were assigned per dialogue, we ordered them by the time they occurred in the transcription.

Two coders ensured reliability, following the advice given by [32]. We performed two-way mixed Intraclass Correlation Coefficient (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 according to [32]. 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 i.e., 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).

*3.4.2 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.

[4]NVivo Website: https://www.qsrinternational.com/nvivo

In addition, after each distributed design session, 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 and discuss the results of this study based on our two RQs.

### 4.1 RQ1: Co-located vs. Distributed Design Thinking

Figures 6 and 7 show the *design thinking* processes of the co-located 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.

Looking at absolute frequencies, we observe a higher number of *design thinking* interactions in the co-located 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.

In terms of relative frequency, we see that all teams in Case 2 have a larger percentage of solution space exploration than any of

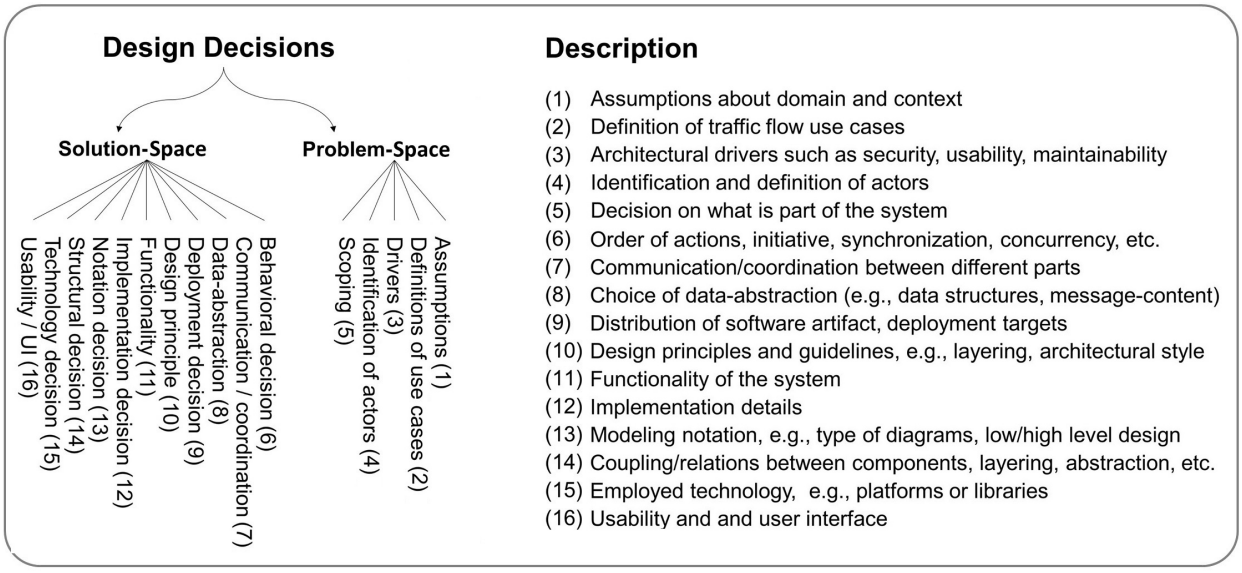


**Figure 5: Classification schema of design decisions into problem space and solution space decisions (based on [48])**

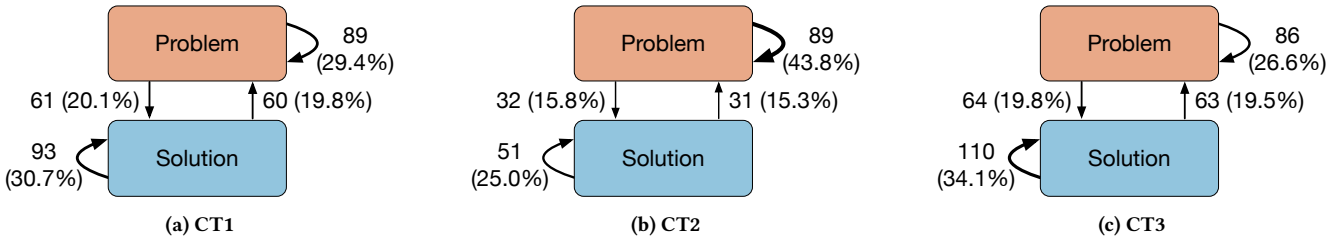


**Figure 6: The *design thinking* of Co-located Teams CT1, CT2, and CT3 (Case 1))**

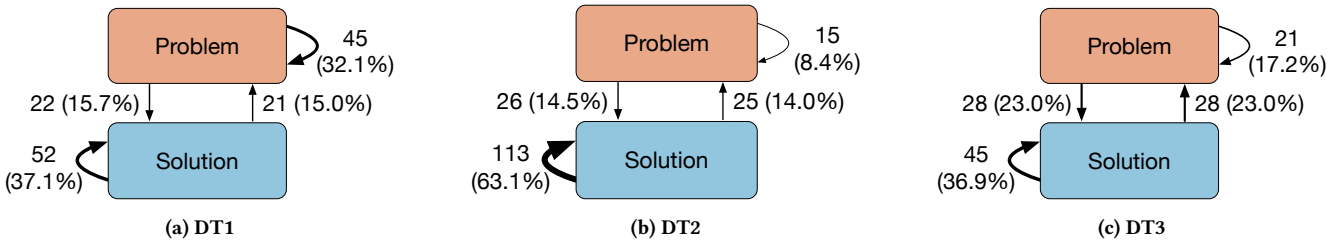


**Figure 7: The *design thinking* of the Distributed Teams DT1, DT2, and DT3 (Case 2)**

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, co-located 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 geographic distance. We summarize this result observation as:

> **Main Observation:** *Geographic distance affects design thinking by reducing problem space exploration, and alignment between problem and solution space.*

Our results provide a potential explanation as to how design communication is hampered in geographically-distributed teams, namely by reducing problem space exploration in favour of a more solution-oriented communication. Clark et al. [10] 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 [12]. 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 distance can hamper communication and coordination, e.g., [3, 14, 23].

In summary, we observe that geographic distribution reduces problem space exploration in collaborative software design tasks compared to the co-located setting, thus answering RQ1. We believe that this finding explains in more detail how communication is affected by geographic distance.

## 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*, i.e., 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, then discuss these results in relation to existing work.

Based on the post-study 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. Gestures or facial expressions were hard or impossible to recognize. This was the case despite the provided video-conferencing facilities, e.g., when they were moving in front of the interactive whiteboard. This is a technological challenge known from related work on distributed work [20]. 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 was a lack of common understanding. This challenge refers to individuals not knowing whether they have the same level of knowledge or understanding of the subject matter as their remote counterpart [23]. In the Gap model [3], this challenge relates to cognitive distance. This issue could be reduced by collaborating with developers from the same department or school of thought.

Two participants mentioned network problems. Furthermore, technology contributes strongly to the awareness challenge, and could additionally contribute to misinterpretations of discussions (reported by one participant).

The results of the post-study questionnaire confirm Proposition B (tool support and social factors are the most frequent challenges in GSE). However, socio-cultural factors might be under-represented, since participants are likely more aware of technical limitations, e.g., due to cognitive bias.

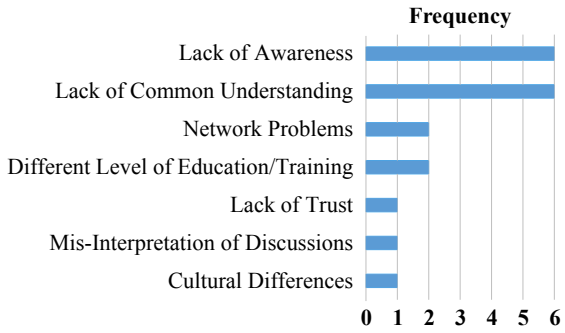


**Figure 8: Perceived challenges to distributed design**

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 [24], 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 socio-technical factors, the Gap model [3] includes artifact-related distances, e.g., 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 co-located 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, e.g., by not exploring the problem space sufficiently (as noted in our **Main Observation**). 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 [9]. This effect might be amplified by socio-technical barriers. Secondly, 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 (R.H.):** *Lack of awareness of the remote counterpart and lack of common understanding reduce the amount of problem space exploration in distributed* design thinking.

# 5 THREATS TO VALIDITY

We identified and grouped the threats to validity in our study according to Yin [51]:

## 5.1 Construct Validity

*How well do operational measures represent what we intended them to represent in the study in question?*

First, *design thinking* is a concrete phenomenon, a natural and ubiquitous human activity during problem-solving processes [11]. We assessed explicit *design thinking*, i.e., *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*). Suppose that tacit *design thinking* happens, then our expectation is that this happens less in Case 2, as geographic distances increase the social challenges which might make the expression of tacit knowledge much harder.

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 observed the distributed design sessions and noted the encountered challenges. This allowed us to triangulate the challenges.

## 5.2 Internal Validity

*Are causal relationships and confounding factors identified and alleviated?*

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 [1, 12]. The results of this study expose and explain the *design thinking* of expert developers. To understand the behaviour of novice developers, we call for replication. However, designers in Case 1 had substantially more experience than designers in Case 2. This is a potential confounding factor that explains the differences in effort spent on problem and solution space exploration.

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. Hence, we do not think that differences in perception affected our results too much.

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 minutes) 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 [5]. The average SUS score of $74.17 \pm 5.63$ (good usability according to [45]) 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.

The design sessions lasted about 90 minutes, i.e., designers 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*. We assume that fatigue would have a similar effect in both cases, and therefore not affect our comparison.

## 5.3 External Validity

*To which extent can the results of our study be generalized?*

By design, case studies have a very limited external validity stemming from the fact that a topic is studied within its context. Therefore, we cannot claim that our findings are generalizable, i.e., 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, e.g., in terms of size, terminology, language, and level of detail. This also limits the generalizability of the findings.

Finally, designers were selected based on contacts, i.e., through convenience sample. Since the study's aim is not to generalize over a population of actors, we do not consider this a major threat.

## 5.4 Reliability

*To which extent can the operations of our study be repeated by other researchers, achieving the same results?*

The case study design and process is described in detail in Section 3. Furthermore, we use the published task description by [39], whose material can be obtained at 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 CONCLUSION

In this paper we reported a multiple-case study exploring the effect of geographic distance on design thinking (RQ1), and the challenges perceived by geographically-distributed teams (RQ2). We observed that distributed developers did less *design thinking* than the co-located developers, specifically in problem space exploration and in the alignment between the problem and solution space (**Main Observation**). Participants in the distributed case perceived a lack of the awareness of the remote counterpart and a lack of common

understanding as the main challenges in distributed design. We hypothesize (**R.H.**) that these perceived challenges might be causing the observed reduction in *design thinking*.

## 6.1 Implications to Research

Our study extends the body of knowledge in GSE, since it offers an explanation how geographic distance affects 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 solution strategies in distributed settings. Similarly, new techniques that specifically target problem space exploration can be proposed. For instance, future work could build on the reminder cards proposed by Tang et al. [46], suggesting a set of reminder cards that foster problem space exploration.

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 (**R.H.**). 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, i.e., how socio-technical 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.

## 6.2 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 co-located teams, e.g., 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, e.g., 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 non-technical factors and can reinforce practitioners in initiatives that aim to mitigate these barriers.

## 6.3 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 socio-cultural dimension in this context, e.g., by controlling for different national cultures or knowledge gaps between different geographic sites.

## REFERENCES


[1] Saeema Ahmed, Ken M Wallace, and Lucienne T Blessing. 2003. Understanding the differences between how novice and experienced designers approach design tasks. *Research in engineering design* 14, 1 (2003), 1–11.

[2] Alex Baker and André van der Hoek. 2010. Ideas, subjects, and cycles as lenses for understanding the software design process. *Design Studies* 31, 6 (2010), 590–613.

[3] Elizabeth Bjarnason, Kari Smolander, Emelie Engström, and Per Runeson. 2016. A theory of distances in software engineering. *Information and Software Technology* 70 (2016), 204–219.

[4] Pernille Bjørn, Morten Esbensen, Rasmus Eskild Jensen, and Stina Matthiesen. 2014. Does distance still matter? Revisiting the CSCW fundamentals on distributed collaboration. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 5 (2014), 27.

[5] John Brooke. 2013. SUS: a retrospective. *Journal of usability studies* 8, 2 (2013), 29–40.

[6] Frederick P Brooks Jr. 2010. *The design of design: Essays from a computer scientist.* Pearson Education.

[7] Richard Buchanan. 1992. Wicked problems in design thinking. *Design issues* 8, 2 (1992), 5–21.

[8] Mauro Cherubini, Gina Venolia, Rob DeLine, and Andrew J Ko. 2007. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 557–566.

[9] Henri Christiaans and Rita Assoreira Almendra. 2010. Accessing decision-making in software design. *Design Studies* 31, 6 (2010), 641–662.

[10] Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. In *Perspectives on socially shared cognition*, L. B. Resnick, J. M. Levine, and S. D. Teasley (Eds.). Vol. 13. American Psychological Association, 127–149.

[11] Nigel Cross. 2001. Designerly ways of knowing: Design discipline versus design science. *Design issues* 17, 3 (2001), 49–55.

[12] Nigel Cross. 2004. Expertise in design: an overview. *Design studies* 25, 5 (2004), 427–441.

[13] Nigel Cross. 2011. *Design thinking: Understanding how designers think and work.* Berg.

[14] Daniela Damian, Sabrina Marczak, and Irwin Kwan. 2007. Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. In *15th IEEE International Requirements Engineering Conference (RE 2007)*. IEEE, 59–68.

[15] Daniela Damian and Deependra Moitra. 2006. Guest editors' introduction: Global software development: How far have we come? *IEEE software* 23, 5 (2006), 17–19.

[16] Uri Dekel. 2005. Supporting distributed software design meetings: what can we learn from co-located meetings? *ACM SIGSOFT Software Engineering Notes* 30, 4 (2005), 1–7.

[17] Franziska Dobrigkeit and Danielly de Paula. 2019. Design thinking in practice: understanding manifestations of design thinking in software engineering. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1059–1069.

[18] Kees Dorst. 2011. The core of 'design thinking' and its application. *Design studies* 32, 6 (2011), 521–532.

[19] Kees Dorst and Nigel Cross. 2001. Creativity in the design process: co-evolution of problem–solution. *Design Studies* 22, 5 (2001), 425–437.

[20] Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work (CSCW âĂŹ92)*. 107âĂŞ–114. https://doi.org/10.1145/143457.143468

[21] Christof Ebert, Marco Kuhrmann, and Rafael Prikladnicki. 2016. Global software engineering: Evolution and trends. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 144–153.

[22] Tracy Hammond, Krzysztof Gajos, Randall Davis, and Howard Shrobe. 2002. An agent-based system for capturing and indexing software design meetings. In *Agents in Design*, J.S. Gero and F.M. Brazier (Eds.). Key Centre of Design Computing and Cognition, University of Sydney, Australia.

[23] James D Herbsleb. 2007. Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering (FOSE'07)*. IEEE, 188–198.

[24] Geert Hofstede. 2001. *Culture's consequences: Comparing values, behaviors, institutions and organizations across nations.* Sage publications.

[25] Rodi Jolak and Grischa Liebel. 2019. Position Paper: Knowledge Sharing and Distances in Collaborative Modeling. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 415–416.

[26] Rodi Jolak, Eric Umuhoza, Truong Ho-Quang, Michel R. V. Chaudron, and Marco Brambilla. 2017. Dissecting design effort and drawing effort in UML modeling. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 384–391.

[27] Rodi Jolak, Boban Vesin, and Michel R. V. Chaudron. 2017. OctoUML: an environment for exploratory and collaborative software design. In *39th International Conference on Software Engineering. ICSE*, Vol. 17. 7–10.

[28] Rodi Jolak, Andreas Wortmann, Michel R. V. Chaudron, and Bernhard Rumpe. 2018. Does Distance Still Matter? Revisiting Collaborative Distributed Software Design. *IEEE Software* 35, 6 (2018), 40–47.

[29] Jeff Wai Tak Kan and JS Gero. 2013. Studing software design cognition. In *Software Designers in Action: A Human-Centric Look at Design Work*, Andre Van Der Hoek and Marian Petre (Eds.). Chapman and Hall/CRC.

[30] Arif Ali Khan, Jacky Keung, Mahmood Niazi, Shahid Hussain, and Mohammad Shameem. 2019. GSEPIM: A roadmap for software process assessment and improvement in the domain of global software development. *Journal of software: Evolution and Process* 31, 1 (2019), e1988.

[31] Lucy Kimbell. 2011. Rethinking design thinking: Part I. *Design and Culture* 3, 3 (2011), 285–306.

[32] Terry K Koo and Mae Y Li. 2016. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine* 15, 2 (2016), 155–163.

[33] Nale Lehmann-Willenbrock, Joseph A Allen, and Annika L Meinecke. 2014. Observing culture: Differences in US-American and German team meeting behaviors. *Group Processes & Intergroup Relations* 17, 2 (2014), 252–271.

[34] Tilmann Lindberg, Christoph Meinel, and Ralf Wagner. 2011. Design thinking: A fruitful concept for IT development? In *Design thinking*. Springer, 3–18.

[35] Nicolas Mangano, Alex Baker, Mitch Dempsey, Emily Navarro, and André van der Hoek. 2010. Software design sketching with calico. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. 23–32.

[36] Roger Martin. 2017. The design of business: why design thinking is the next competitive advantage.

[37] Eoin Ó Conchúir, Helena Holmström Olsson, Pär J Ågerfalk, and Brian Fitzgerald. 2009. Benefits of global software development: exploring the unexplored. *Software Process: Improvement and Practice* 14, 4 (2009), 201–212.

[38] Gary M Olson and Judith S Olson. 2000. Distance matters. *Human–computer interaction* 15, 2-3 (2000), 139–178.

[39] Marian Petre and Andre Van Der Hoek. 2013. *Software Designers in Action: A Human-Centric Look at Design Work*. Chapman and Hall/CRC.

[40] Marian Petre, André van der Hoek, and David S Bowers. 2019. Software design as multiple contrasting dialogues. In *Psychology of Programming Interest Group 30th Annual Conference, 28-30 Aug 2019, Newcastle University*. 8.

[41] Marian Petre, André van der Hoek, and Yen Quach. 2016. *Software Design Decoded: 66 Ways Experts Think*. MIT Press.

[42] Maryam Razavian, Antony Tang, Rafael Capilla, and Patricia Lago. 2016. In two minds: how reflections influence software design thinking. *Journal of Software: Evolution and Process* 28, 6 (2016), 394–426.

[43] Horst WJ Rittel and Melvin M Webber. 1973. Dilemmas in a general theory of planning. *Policy sciences* 4, 2 (1973), 155–169.

[44] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. 2012. Case study research in software engineering. In *Guidelines and examples*. Wiley Online Library.

[45] Jeff Sauro. 2011. *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC Denver, CO.

[46] Antony Tang, Floris Bex, Courtney Schriek, and Jan Martijn EM van der Werf. 2018. Improving software design reasoning–a reminder card approach. *Journal of Systems and Software* 144 (2018), 22–40.

[47] Thomas S Tullis and Jacqueline N Stetson. 2004. A comparison of questionnaires for assessing website usability. In *Usability professional association conference*, Vol. 1. Minneapolis, USA, 12.

[48] Rainer Weinreich, Iris Groher, and Cornelia Miesbauer. 2015. An expert survey on kinds, influence factors and documentation of design decisions in practice. *Future Generation Computer Systems* 47 (2015), 145–160.

[49] Courtney Williams, Margaret-Anne Storey, Neil A Ernst, Alexey Zagalsky, and Eirini Kalliamvakou. 2019. Methodology Matters: How We Study Socio-Technical Aspects in Software Engineering. *arXiv preprint: 1905.12841* (2019).

[50] James Wu and TC Nicholas Graham. 2004. The software design board: a tool supporting workstyle transitions in collaborative software design. In *IFIP International Conference on Engineering for Human-Computer Interaction*. Springer, 363–382.

[51] Robert K Yin. 2017. *Case study research and applications: Design and methods*. Sage publications.