Noname manuscript No.
(will be inserted by the editor)

Software Engineering Whispers:

The Effect of Textual Vs. Graphical Software Design Descriptions on Software Design Communication

Rodi Jolak · Maxime Savary-Leblanc · Manuela Dalibor · Andreas Wortmann · Regina Hebig · Juraj Vincur · Ivan Polasek · Xavier Le Pallec · Sébastien Gérard · Michel R.V. Chaudron

Received: date / Decision: date

Abstract *Context*: Software engineering is a social and collaborative activity. Communicating and sharing knowledge between software developers requires much effort. Hence, the quality of communication plays an important role in influencing project success. To better understand the effect of communication on project success, more in-depth empirical studies investigating this phenomenon are needed. *Objective*: We investigate the effect of using a textual versus graphical software design description on co-located software design communication. *Method*: We conduct a family of experiments involving 240 software engineering students from four universities. We examine how different software design representations (i.e., graphical vs. textual) influence the ability to *explain*, *understand*, *recall*, and *actively communicate* knowledge. *Results*: We find that the graphical software design description is better than the textual description in promoting *active discussion* and supporting the *recall* of design details. Furthermore, compared to its unimproved version, a *well-organized* and *motivated* textual design description (that is used for the same amount of time) enhances the *recall* of design details and increases the amount of *active discussions* at the cost of reducing the perceived quality of *explaining*.

Keywords Software Engineering · Software Design · Software Modeling · UML · Communication · Knowledge Sharing · Graphical Representation · Textual Representation · Family of Experiments

1 Introduction

Software engineering is a social activity and requires intensive communication and collaboration between developers. In large companies, developers work in different development teams and collaboratively communicate with many

R. Jolak
E-mail: rodi.jolak@cse.gu.se

Extended author information available on the last page of the article.

stakeholders. In such a setting, the quality of communication between the stakeholders plays an important role in reducing the overall teams', and thus projects', development effort. In a multiple-case study on challenges and efforts of model-based software engineering approaches, Jolak et al. [30] analyzed the distribution of efforts over different development activities in two software engineering projects. Interestingly, they found that communicating and sharing knowledge dominates the effort spent by developers. The effort on communication, as Jolak et al. found, is actually more than all of the efforts that developers spent in any of the other observed development activities, such as, requirements analysis, design, coding, testing, integration, and deployment.

Furthermore, poorly defined software applications (due to miscommunication between stakeholders) can affect the final structure and/or behavior of these applications. This is in line with Jarboe et al. [28] and Kortum et al. [35] who consider that the quality of communication does influence developers' activity experience and achievement, and therefore customer's satisfaction.

The aforementioned studies underline the importance of communication in Software Engineering (SE). They also highlight the need to study communication in-depth to determine elements or criteria of its efficiency and effectiveness. The study we present in this article is inline with this concern: we investigate how different software architecture design representations affect the communication of design knowledge. In particular, we compare textual vs. graphical representations. In contrast to a *textual* representation, a *graphical* representation provides a two-dimensional visuospatial description of information reflecting the actual spatial configurations of the parts of a process or system [59]. With respect to knowledge communication, we look into the following *communication* aspects:

1. *Explaining*: or knowledge donating, communicating the personal intellectual capital from one person to others [12].
2. *Understanding*: or knowledge collecting, receiving others' intellectual capital [12].
3. *Recall*: or memory recall, recognizing or recalling knowledge from memory to produce or retrieve previously learned information [1].
4. *Collaborative Interpersonal Communication* [53], which includes:
   (a) *Active Discussion*: questioning, informing, and motivating others.
   (b) *Creative Conflict*: arguing and reasoning about others' discussions.
   (c) *Conversation Management*: coordinating and acknowledging communicated information.

1.1 Rationale

Kauffeld et al. [33] suggested that effective team communication and information flow are prerequisites for the success of software development projects. In a study on requirements practices in start-ups, Gralha et al. [19] identified knowledge management and communication as increasingly important strategies for risk mitigation and prevention. As a consequence, research concerning different factors influencing the degree and way in which people communicate and share their knowledge is actually relevant for maximizing the aforementioned advantages.

Graphical descriptions encode and present knowledge differently from textual descriptions. In particular, they provide a visuospatial representation of information, and can recraft these information into a multitude of forms by using fundamental graphical elements, such as dots and lines, nodes and links [59]. Moreover, graphical descriptions encourage spatial inferences (e.g., inferences about the behavior, causality, and function of a system) to substitute for and support abstract inferences [3]. This is inline with Moody [41], who states that graphical and textual knowledge representations are differently processed by the human mind. Empirical evidence on how graphical descriptions affect developer's achievement and development productivity is still underwhelming, as reported by Hutchinson et al. [27]. Moreover, Meliá et al. [40] report that the software engineering field lacks a body of empirical knowledge on how different representations (graphical vs. textual) could provide support for improving software quality and development productivity.

In this study, we focus on design knowledge communication/transfer between two software developers, where, by using a graphical vs. textual software design description, one developer is taking the role of design *Explainer* (i.e., design knowledge owner), and one developer is taking the role of design *Receiver* (i.e., design knowledge receiver). Rus et al. [48] reported that greatest challenge of companies is to retain tacit knowledge, mainly, but also explicit knowledge (e.g., models).

Companies, such as *Ericsson Software Technology*[1] and *sd&m*[2], started initiatives –Ericsson's initiative is called "*Experience Engine*"– to exchange knowledge between developers by connecting two individuals, a *problem owner* and *experience communicator*. The problem owner is the employee who requires information or support to solve a specific problem and the experience communicator is the employee who has in-depth knowledge of the problem domain. Having been connected, the experience communicator has to *educate* the problem's owner on how to solve it. The aforementioned initiatives illustrate that our study has a practical relevance.

1.2 Objective and Contribution

We planned and conducted a family of experiments with a goal to *understand* and *compare* the effect of using a Graphical Software Design description (GSD) *versus* a Textual Software Design description (TSD) on software design communication. Through this, we contribute to the body of empirical knowledge on the practical use of graphical versus textual software design descriptions. Such knowledge might lead to achieving more effective software design communication, which in turn would help in reducing the total effort of software development activities. Consequently, we address the research objective by answering the following question:

[1] `https://www.ericsson.com`

[2] `https://www.capgemini.com`

- **R.Q.1** How does the representation of software design (graphical vs. textual) influence *[Communication Aspect]*?

Where the investigated *[Communication Aspect]s* are the following: (1) Design Explaining, (2) Design Understanding, (3) Design Recall, (4) Active Discussion, (5) Creative Conflict, and (6) Conversation Management.

We first *understand* how each software design representation (i.e., graphical/textual) affect the six aspects of communication that we described previously (i.e., explaining, understanding, recall, active discussion, creative conflicts, and conversation management). Then, we *compare* the effect of using the graphical vs. textual software design description on the considered communication aspects.

To address certain threats to external validity, we also compare the effect of using a *cohesive*[3] and *motivated*[4] TSD versus less cohesive and unmotivated TSD on software design communication. In particular, we address the following research question:

- **R.Q.2** Does using a cohesive and motivated TSD influence *[Communication Aspect]*?

Where the investigated *[Communication Aspect]s* are the following: (1) Design Explaining, (2) Design Understanding, (3) Design Recall, (4) Active Discussion, (5) Creative Conflict, and (6) Conversation Management.

The remainder of this paper is organized as follows: We discuss the related work in Section 2. We describe the family of experiments in Section 3. We present the results in Section 4. We discuss the results and threats to validity in Section 5. Finally, we conclude and describe the future work in Section 6.

2 Related Work

Effective communication depends on various factors, such as personality [10], distance [31], or knowledge representation (graphical vs. textual) [23,40,50].

In a recent study on design activities of co-located and distributed collaborative software design [31], the authors investigate whether advanced technologies for distributed communication can replace personal meetings. The main result is that co-located face-to-face meetings remain relevant as facial reactions and body language are often not transmitted by current communication software. This is partly due to technical challenges, such as unstable or slow Internet connection, that affect communication results. In contrast to that study, our family of experiments does not investigate distributed communication and is conducted without the use of communication tool-support to mitigate the effects of technical challenges.

Meliá et al. [40] describe an experiment in which students perform maintenance tasks on a graphical model and on a textual model. The authors investigate whether a model's syntax affects subjective and objective performance

[3] *Cohesive*: documented information or knowledge that are well-organized.

[4] *Motivated*: augmented with design rationale.

and whether the notation influences developer satisfaction. Objective performance is measured by the number of correct answers in the task whereas the subjective performance is the performance as perceived by the developer. In the experiment, participants were divided into two groups, one group worked with a model for selling tickets, the other group had a model for organizing online courses. Participants received the models in textual and in a graphical notation and were asked to find 5 errors in each notation. They also received 5 tasks in which they had to extend and modify each model. Participants using the textual notation performed significantly better in finding errors in the domain model and also spent less time until finishing the task. Nonetheless, participants preferred to work with the graphical notation. The authors believe this to originate from the fact that students learn graphical modelling languages such as UML as stereotypes for domain models whereas less attention is given to textual modelling languages.

In another study, the authors measure how well participants extract the required information (such as architectural design decisions) from different media [23]. The researchers collected participant-specific information in two questionnaires, filmed participants during tasks, and asked them to think out loud. The experiment is comprised of four architectures, out of which each consisted of a graphical and a textual description. Participants (students and professional developers) were asked three questions per architecture. The authors observed that no notation was clearly superior in communicating architecture design decisions. Nonetheless, participants tended to first look at the graphical notation before reading the text. The authors attribute this to the clarity of the graphic representation, which enables participants to grasp the structure of the model more quickly.

In a case study on comparing graphical versus textual representations, the researchers measure the accuracy and time spent to solve three requirement comprehension tasks [50]. The study does not indicate results concerning accuracy (both notations yield correct results), but participants spent less time when working with textual requirements. Participants preferred to work with the graphical representation nonetheless. Also, when working with a combination of graphical and textual representations, the study measured the best results concerning time and accuracy.

Other research investigated a combined usage of textual and graphical representations [37]. The researcher interviewed 21 practitioners to find out how developers work with different requirements artifacts of various granularity and notation and how they handle scattered information. The researcher found out which artifacts the practitioners used and which problems they encountered. A share of 70% of the interviewees reported issues when working with multiple artifacts. The main shortcomings were inconsistencies and the additional effort for documenting.

In contrast to our research, the studies described in [23,37,40,50] do not observe communication based on using graphical and textual representations, but how well both notations are suited to share information. Therefore, more research concerning both notations as a base for explaining and discussing software architectures is required.

Other related research aims to find out if drawing improves the recall ability, compared to making textual notes [39]. The participants of this research were divided into two groups, younger and older adults, to measure whether the notation influences both groups in the same way. Participants were told 30 nouns, one after the other, and asked to either draw or to note the items textually. Afterwards, they were asked to recall and list all items. For the drawn nouns both groups performed equally well, but for the textual words, young adults performed better. This indicates that a graphical notation can compensate for the age-related deficit.

To summarize the related work section, Table 1 provides a brief comparison between the research objectives of our study and related work.

Table 1: Research objectives of our study and related work.

Work	**Objectives**
Heijstek et al. [23]	Study the effect of using a graphical vs. textual on extracting design decisions information.
Jolak et al. [31]	Study the effect of distance on communication
Liskin [37]	Study the use of different requirement artifacts in practice.
Meade et al. [39]	Study the effect of drawings vs. textual notes on memory recall.
Meliá et al. [40]	Study the effect of using a graphical vs. textual notation on domain models maintenance.
Sharafi et al. [50]	Study the effect of using a graphical vs. textual representation on requirement comprehension.
Our Study	*Study the effect of using a graphical vs. textual software design description on face-to-face design communication.*

3 Experimental Design

This section describes the protocol that is used to perform the experiments and analyze the results. In particular, we report the experiment according to the guidelines suggested by Jedlitschka et al. [29].

3.1 Family of Experiments

Easterbrook et al. [15] highlighted that controlled experiments help to investigate testable hypotheses where one or more independent variables are manipulated to measure their effect on one or more dependent variables. A family of experiments facilitates building knowledge and extracting significant conclusions through the execution of multiple experiments pursuing the same goal. Basili et al. [2] reported that families of experiments can help to (i) generalize findings across studies and (ii) contribute to important and relevant hypotheses that may not be suggested by individual experiments.

We planned and conducted a family of experiments based on the methodology of Wohlin et al. [62]. Our family of experiments are between-subject designs to minimize learning effects and transfer across conditions. The family of experiments consists of one original experiment and three external replications involving 240 participants in total (See Table 2). The original experiment (OExp) was conducted at the University of Gothenburg involving a mix of 50 B.Sc. and M.Sc. Software Engineering students. In OExp, we study the effect of using a graphical software design description (GSD) vs. textual software design description (TSD) on software design communication. The first replication (REP1) was conducted at RWTH Aachen University with 36 M.Sc. and Ph.D. SE students. The second replication (REP2) was conducted at the University of Lille involving 94 M.Sc. SE students. REP1 and REP2 replicated the original experiment as accurately as possible (*strict replications* [2]). The third replication (REP3) was conducted at the Slovak University of Technology with 60 B.Sc. and M.Sc. SE students. REP3 varied the manner in which the original experiment was conducted, so that certain threats to external validity were addressed. More specifically, REP3 is a replication that varies a variable intrinsic to the object of study [2]: we study the effect of using a graphical (GSD) vs. altered- textual software design description (A-TSD) on software design communication. More details regarding this variation are provided in Section 3.5.

The experiment material and communication language in OExp, REP1, and REP3 were in *English*. In contrast, the experimental material and communication language in REP2 (which was conducted at the University of Lille, France) were in *French*. The gender distribution in each experiment is also shown in Table 2. The majority of the participants are males (79%).

Table 2: The family of experiments.

ExpID (R.Q.)	**S.R.**	**C.D.**	**Context**	**Lang.**	**Date**	**Participants**	**#**	**Females**
OExp (R.Q.1)	-	GSD vs. TSD	Chalmers & Gothenburg University	English	11/10/18	B.Sc. & M.Sc. Students	50	22%
REP1 (R.Q.1)	Yes	GSD vs. TSD	RWTH Aachen University	English	25/10/18	M.Sc. & Ph.D. Students	36	17%
REP2 (R.Q.1)	Yes	GSD vs. TSD	University of Lille	French	03/12/18	M.Sc. Students	94	26%
REP3 (R.Q.2)	No	GSD vs. A-TSD	Slovak University of Technology	English	13/12/18	M.Sc. Students	60	15%
						Total	240	21%

R.Q.: Research Question
S.R.: Strict Replication
C.D.: Compared Designs

3.2 Scope

Developers intensively communicate ideas, decisions, progress, and updates throughout the software development life-cycle. In this study, we focus on investigating co-located, face-to-face software design communication.

Design communication plays a fundamental role in transferring software design decisions (i.e., instructions for software construction) from *architects/analysts* to *programmers* or other stakeholders. Also, the quality of these communications might play an important role in shaping the overall structure and behaviour of software products.

In co-located teams, developers usually communicate face-to-face. In distributed teams, developers use other communication channels, such as video conferencing systems. Jolak et al. [31] found that co-located software design discussions are more effective than distributed design discussions. Moreover, Storey et al. [54] stated that face-to-face communication is one of the most important and preferred communication channel for collaborative software development. Indeed, with face-to-face communications, developers can receive feedback quickly which facilitates discussing through complex issues, such as design decisions. Thus, we investigate co-located face-to-face communication, as this type of communication is widely preferred and would therefore contribute to the generalizability of our results.

Modeling languages can be (i) of general-purpose and applied to any domain, such as the Unified Modeling Language (UML) or (ii) domain-specific and designed for a specific domain or context, such as the Domain Specific Languages (DSLs). Brambilla et al. [5] stated that UML is widely known and adapted, and comprises a set of different diagrams for describing a system from different perspectives. Brian Dobing and Jeffrey Parsons [13] found that the use of UML class diagrams substantially exceeds the use of any other UML diagram (use case, sequence, activity, etc.). Thus, in order to increase the generalizability of the results of this study we chose to represent the graphical software design description by a UML class diagram.

3.3 Participants

The population for this study was intended to match two prerequisites: (i) having a basic knowledge in UML (especially UML class models), (ii) and being able to understand and communicate in the experiment language. The target group in this case is the entire group of people who posses the aforementioned criteria: students who took an academic course in UML modeling, professional software developers, architects, etc. However, the portion of the population to which we had reasonable access is a subset of the target population. In particular, the accessible population for this family of experiments was the group of B.Sc. and M.Sc. Software Engineering (SE) students at the universities where the authors teach SE courses. The sampling approach was *convenience sampling*. On the one hand, this sampling approach is easy and readily available. On the other hand, the sample produced by convenience sampling might not represent the entire population (i.e., threat to external validity or generalizability of the results). To increase the external validity of the results, we recruited a mix of *240* B.Sc. and M.Sc. SE students from four universities to take a part in a family of experiments. Previously in Section 3.1 (Table 2), we provided details on the participants in this family of experiments.

3.4 Experimental Treatments

The participants of each experiment were *randomly* assigned to two treatments or groups:

- **Group G**: participants in this group had to discuss a software design as represented by a graphical description (UML class diagram).
- **Group T**: participants in this group had to discuss the same software design, but as represented by a textual description.

Furthermore, the participants of each group were *randomly* assigned one specific role:

- **Explainer**: this role consisted in: (i) understanding the design representation, and (ii) explaining it to a *Receiver*.
- **Receiver**: this role consisted in understanding the software design based on the discussion with an *Explainer*.

Having the roles assigned, we *randomly* formed 120 *Explainer-Receiver* pairs. These pairs were involved in discussing a design case which we detail in the next Section 3.5.

3.5 Design Case and Graphical vs. Textual Descriptions

We created a design case for our family of experiments. The design case describes a structural view of a mobile application of a fitness center, the *Fitness Paradise*. This fitness center gives its clients the opportunity to book facilities and activities. The featured application enables clients to consult the schedule of activities, manage bookings, keep track of payments, and visualize performance data when available. We believe that the selected design case relies on a familiar domain, *Sport and Gym*, from everyday life which is quite popular and easy to understand without prior knowledge.

To introduce the *Explainers* with the design case, we created a *design case specification*[5] document which describes the fitness center and lists the features of the mobile application in natural language.

3.5.1 Design Descriptions

We played the role of the *designer/architect* and created the two design representations (i.e., GSD and TSD). The GSD and TSD provide the same information and describe one structural design of the *Fitness Paradise* app. The two design descriptions only differ in the way they represent the design (i.e., graphical vs. textual).

- **GSD**. We created the UML class diagram[6] of the design case. The diagram includes 28 classes (21 model entities, 3 controllers, and 4 views)

[5] `http://rodijolak.com/SE_Whispers/Design_Case.pdf`

[6] `http://rodijolak.com/SE_Whispers/GSD.pdf`

and 30 relationships. We chose to use the Model View Controller (MVC) design pattern for structuring the design, as this pattern is well known by the participants of the experiments. The entities of each part of the MVC were given a specific color. The model entities have a yellow color, the controllers are in blue, and the views are in green. The colors were added to the entities in the GSD in order to mimic the characteristic of structured textual document (which we describe next) in facilitating a visual distinction between different sections (i.e., the three parts of the MVC).

- **TSD**. For EXP, REP1, and REP2, each element of the GSD was systematically used to create exactly one corresponding element (e.g., one paragraph or sentence) in the textual description, thus to maintain a one-to-one correspondence between GSD and TSD. In other words, we were thoroughly keen to make both the graphical and textual designs present the exact same amount of information or design knowledge in order to control eventual bias due to a different amount of information. The textual description[7] was arranged into two main structured sections. In the first section, we orderly described the entities of each module of the MVC: First the entities of the model part, then the entities of the controller part, and last the entities of the view part. In the second section, we described the relationships between the entities following the same appearance order of the entities.
- **Altered TSD**. In REP3, we used an *altered TSD*[8] to know whether or not a *motivated* and *cohesive* TSD could affect design communication differently from the original TSD.
  (i) *Motivated*: We added an introduction to the original textual description, including a rationale of the chosen design pattern (i.e., MVC).
  (ii) *Cohesive*: The textual design description was organized differently. In particular, the description of the relationships of each entity was moved and placed right after the description of the entity. In this way, information regarding each entity and its relationships are close to each other, instead of being distant/remote (i.e., located on different pages), as in the original textual description.

3.6 Tasks

The main task of this family of experiments was inspired by the Chinese Whispers (or The Telephone) game. In this game, players form a line, and the first player comes up with a message and whispers it to the ear of the second person in the line. The second player repeats the message to the third player, and so on. When the last player is reached, they announce the message they heard to the entire group.

In contrast, we created a message (i.e., a software design representation), and asked the first player (i.e., the *Explainer*) to first understand the mes-

[7] `http://rodijolak.com/SE_Whispers/TSD.pdf`

[8] `http://rodijolak.com/SE_Whispers/Altered_TSD.pdf`

sage then explain it to the second player (i.e., the *Receiver*). After that, the players (i.e., *Explainers* and *Receivers*) have to announce the message (i.e., via answering a post-task questionnaire). Finally, the original message (i.e., the software design representations) is compared with the final version (i.e., knowledge of *Explainers* and *Receivers*).

The main task of the experiments reflects common scenarios in software engineering industry where developers collaborate, communicate, and exchange knowledge in order to create software. For example, the main task reflects a common knowledge-transfer scenario between a software architect (i.e., Explainer) who owns knowledge on the structure and behavior of the software system and a software developer (i.e., Receiver) who needs to receive and understand the knowledge of the architect in order to start coding. Moreover, knowledge communication is especially important when new employees enter a company and *struggle* to learn the existing tacit knowledge. In this direction, our task reflects the scenario of onboarding of novice developers by experienced developers (e.g., Ericsson's "*Experience Engine*" initiative (cf. section 1)).

In addition to the main task, we added two secondary tasks to collect complementary data, such as participants' design experience and communication skills that are also needed for data analysis and results' discussion.

For the study, the participants had to perform the following three tasks:

1. **Answer the pre-task questionnaire**. All participants have to answer the pre-task questionnaire based on the group they are assigned to. No time-limit is imposed for this task. We noted the required time for this step during the experiments and found that it takes 15 minutes on average.
   The pre-task questionnaire is developed to collect participants' design experiences and communication skills based on self-evaluations. The questions in the pre-task questionnaire vary according to the role of the participant (*Explainer* vs. *Receiver*) and his/her group (G vs. T).
   - **G-Explainer**: participants belonging to this subset have to answer questions on (i) familiarity with software design and UML modeling, (ii) how good are they in understanding and *sense-making*[9] of UML models, an English/French conversation, and explaining their knowledge to others.
   - **G-Receiver**: participants belonging to this subset have to answer questions on (i) familiarity with software design and UML modeling, (ii) how good are they in understanding and sense-making of UML models, an English/French conversation, and *building* knowledge from conversing with others.
   - **T-Explainer**: participants belonging to this subset have to answer questions on (i) familiarity with software design, (ii) how good are they in reading, understanding, and sense-making of an English/French text, understanding and sense-making of an English/French conversation, and explaining their knowledge to others.

[9] Developing the understanding of a concept by connecting it with existing knowledge

   - **T-Receiver**: participants belonging to this subset have to answer questions on (i) familiarity with software design, (ii) how good are they in understanding and sense-making of an English/French conversation, and *building* knowledge from conversing with others.

2. **Discuss the Design (i.e., transfer design knowledge)**. Each *Explainer* receives a *design case specification* plus either a GSD or TSD, based on *Explainer*'s group (G or T). The *Explainer* has to read and understand the received artifacts, as good as he/she can, in 20 minutes (defined based on to the pilot studies, see Section 3.7.1). The *Explainers* are allowed to individually ask questions to experiment supervisors to clarify issues related to the design, if required.
   After 20 minutes, the *Explainers* give the *design case specification* back to the supervisors, but keep the design description (GSD or TSD). Each *Explainer* is randomly paired with a *Receiver* from the same group. Then, each *Explainer-Receiver* pair is given 12 minutes (defined based on to the pilot studies, see Section 3.7.1) to discuss the design, where the *Explainer* has to explain the design and the *Receiver* has to understand the design. The *Receivers* can unhesitatingly ask questions. Moreover to help the understanding process, *Receivers* are allowed to take notes during the discussion, but all notes are collected by the supervisors before the next task. This is because two of the communication aspects, *Understanding* and *Recall*, that we measure require the participants to, respectively, *apply* and *remember* the design knowledge without using the design descriptions or the notes that they took during the discussions.

3. **Answer the post-task questionnaire**. All participants have to answer the post-task questionnaire based on their groups. No time-limit is imposed for this task. We also noted the required time for this step and found that it takes 15 minutes on average.
   The first part of the post-task questionnaire is developed to collect participants' self-evaluations of their experiences just after the design discussion. The questions vary according to the role of the participant (*Explainer* vs. *Receiver*) and his/her group (G vs. T).
   - **G-Explainer**: participants belonging to this subset have to answer questions on (i) how good they are in remembering UML models, (ii) how well they did understand and explain the design, and (iii) how much diagrams did help them in understanding and explaining the design.
   - **G-Receiver**: participants belonging to this subset have to answer questions on (i) how good they are in remembering UML diagrams, (ii) how well they did understand the design from the discussion with the *Explainer*, and (iii) how much diagrams did help them in understanding the design.
   - **T-Explainer**: participants belonging to this subset have to answer questions on (i) how good they are in remembering a English/French text, (ii) how well they did understand and explain the design, and (iii)

in case they could have used them, how much diagrams would have helped in understanding and explaining the design.

- **T-Receiver**: participants belonging to this subset have to answer questions on (i) how good they are in remembering a English/French text, (ii) how well they did understand the design from the discussion with the *Explainer*, and (iii) in case they could have used them, how much diagrams would have helped in understanding the design.

The second part of the post-task questionnaire evaluates participants' understanding and recall abilities.
To measure the *Recall*, we formulated ten questions[10] challenging participants' recall abilities. Two of these questions are open requiring free-text answers, six questions are multiple-choice questions which require the participants to choose only one choice, and two questions are check-boxes questions which require to select one or more answers from the available.
To measure the *Understanding*, we formulated three questions[11] focusing on MVC design maintenance (using maintenance questions to measure understanding is motivated in Section 4.3). In each question we introduce a design maintenance (i.e., evolution) scenario and suggest four ways to address it. The three questions are multiple-choice questions which require the participants to choose only one choice from 4 provided choices.
To evaluate the answers of the participants on recall and understanding questions, we defined grading rules that can be consulted online[12].

Remark. In REP2 (University of Lille), the pre- and post-task questionnaires, design case specification, GSD, and TSD were translated to *French*, as the SE course that the participants are frequenting is in French. During the translation process, each word was carefully chosen to match the semantics of the original *English* textual description as close as possible. To maintain a strict replication, after the translation process we thoroughly did review the aforementioned artifacts and ensured that the amount of information/knowledge they convey is the same as provided by the artifacts used in the original experiment (OExp).

3.7 Variables and Hypotheses

The goal of this study is to compare between the effect of using GSD versus TSD on software design communication. The only independent variable and manipulated factor is the design description. This variable is nominal and corresponds to two treatments/groups: G group using GSD and T group using TSD. In this study, we consider six dependent variables (See Table 3). These variables correspond to the six communication aspects which we described in the introduction.

[10] http://rodijolak.com/SE_Whispers/Recall_Q.pdf

[11] http://rodijolak.com/SE_Whispers/Understanding_Q.pdf

[12] http://rodijolak.com/SE_Whispers/Grading_Rules.pdf

Table 3: Dependent variables and measurement.

Dependent Variable	Measure	Source	Measurement Instrument	Measurement Scale
Explaining (EXP)	Ordinal	Subjective	2 Questions (Perceptions)	5-point Likert Scale Very Poor - Very Good
Understanding (UND)	Interval	Objective	3 Maintenance Questions	Total Score from 0 to Max 3 Points
Recall (REC)	Interval	Objective	10 Recall Questions	Total Score from 0 to Max 10 Points
Active Discussion (AD)	Ratio	Objective	Counting Occurrences in Recorded Conversations	AD/(AD+CC+CM) Values from 0 to 1
Creative Conflict (CC)	Ratio	Objective	Counting Occurrences in Recorded Conversations	CC/(AD+CC+CM) Values from 0 to 1
Conversation Mgt. (CM)	Ratio	Objective	Counting Occurrences in Recorded Conversations	CM/(AD+CC+CM) Values from 0 to 1

The original experiment and replications were conducted under the same environment conditions and by following a well-defined protocol to ensure that the impact of any other variable on the results is relatively negligible.

For OExp, REP1, and REP2, we formulate and study the following *null* $\mathbf{H}_0$ and alternative hypotheses $\mathbf{H}_0^a$:

- $\mathbf{H}\text{EXP}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on EXP.
- $\mathbf{H}\text{UND}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on UND.
- $\mathbf{H}\text{REC}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on REC.
- $\mathbf{H}\text{AD}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on AD.
- $\mathbf{H}\text{CC}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on CC.
- $\mathbf{H}\text{CM}_0^a$: The design description [*has no impact*]$_0$/[has impact]$_a$ on CM.

REP3 varies one variable intrinsic to the object of study (i.e., the independent variable TSD). Accordingly, we study the following *null* $\mathbf{H}_0$ and alternative hypothesis $\mathbf{H}_0^a$:

- $\mathbf{H}\text{TSD}_0^a$: A motivated and cohesive TSD [*has no impact*]$_0$/[has impact]$_a$ on the communication aspect.

3.7.1 Experiment Procedure

Before presenting the experiment procedure, we would like to highlight that we conducted several pilot studies, 2 in the university of Gothenburg, 1 in Aachen university, 1 in Lille university, and 1 in the Slovak university. To cover the treatments of our study, each pilot study involved 2 *Explainer*-*Receiver* pairs (B.Sc., M.Sc., or PhD students in SE). One pair was assigned to the G group using a graphical design description, and the second pair was assigned to the T group using a textual design description. These pilot studies helped us in:

- Designing a research protocol and assessing whether or not it is realistic and workable, especially in estimating the time that is required by: (i) the *Explainer* to understand the design (20 minutes), and (ii) the *Explainer* and *Receiver* to discuss the design (12 minutes).

- Identifying logistical problems and determining what resources (e.g, supervisors, software, and rooms) are needed for the actual experiments.
- Training the supervisors of the experiments.

The experiment procedure was created to define the process of the experiment and to ensure strict replications of the original experiment. Figure 1 presents the four main steps of the experimentation procedure:

- **Step 1**: To anonymize their identity and thus their answers, all participants were randomly assigned an identification number (ID). We asked the participants to bring their PCs to be able to answer the online pre- and post-task questionnaires. *Eduroam Internet* connection was available in the rooms where the experiments were running. Also, we asked the participants to bring a device to record the discussions (either by downloading audio-recording software on their PCs or by using a smart-phone with a recording application). We booked large university lecture-rooms which can host all *Explainer-Receiver* pairs with a sufficient distance between each pair. This helps to reduce voice interference to a minimum and produce better-quality audio recordings. We randomly assigned the participants to two groups (G and T). Furthermore, we randomly assigned each participant one role, *Explainer* or *Receiver*. After that, we asked the participants to answer the pre-task questionnaire.
- **Step 2**: Once all participants filled the pre-task questionnaire, the *Explainers* were taken to a second room where they received the design case specification and the design description (GSD or TSD). The *Explainers* were asked to understand the design that they received as good as they can in 20 minutes. During this time, the *Receivers* ensured that their recording software/devices were working as expected.
- **Step 3**: After 20 minutes, we took the design case specification from the *Explainers*, but let them keep the assigned design description (GSD or TSD). The *Explainers* and *Receivers* were randomly grouped in pairs in one or two rooms according to the number of participants and room's capacity. The pairs were then informed that, using the design descriptions (GSD or TSD), *Explainers* should explain the design to the *Receivers* in 12 minutes. We also informed the pairs that *Receivers* can ask clarification questions to the *Explainers*. When all participants were prepared, we asked the participants to start the audio recorder software and begin the discussions by introducing themselves (by mentioning the name/ID and role). This allowed us later to match the discussion records of the participants with their corresponding answers to the questionnaires.
- **Step 4**: After 12 minutes, the participants were informed that they should stop the audio recording. All documents, including *Receivers*' draft notes, were collected. Then, we asked all the participants to answer the post-task questionnaire individually. Lastly, we asked the participants to rename the audio recordings with their ID numbers and put the recordings in a USB flash drive that we provided.

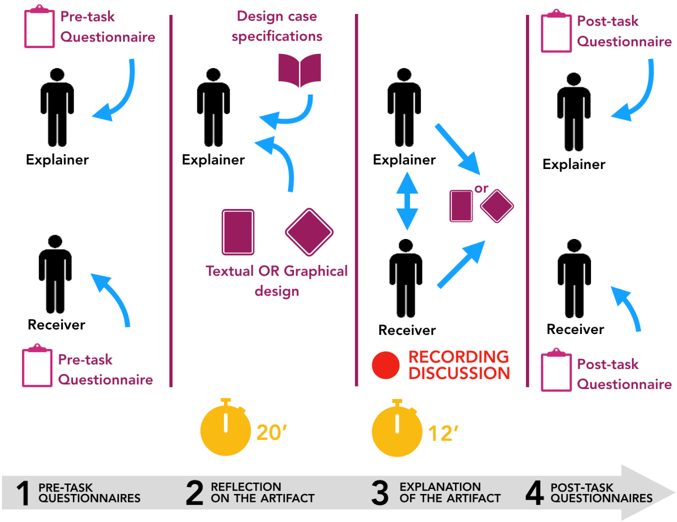


Fig. 1: The four main steps of the experimentation procedure.

3.8 Data Analysis

The data of this study was collected via questionnaires and by audio-recording discussions between *Explainers* and *Receivers*. In this section, we describe three types of analysis procedures that we used:

- *Data Set Preparation*: To check and organize data collected from different sources and prepare it for analysis.
- *Descriptive Statistics*: To describe the basic features of the data by summarizing and showing measures in a meaningful way such that patterns might emerge from the data.
- *Hypothesis Testing*: To make statistical decisions by evaluating two mutually exclusive statements about a population and determining which statement is best supported by the sample data.
- *Meta-Analysis*: To obtain a global effect of a factor on a dependent variable by combining the effect size of different experiments, as well as assessing the consistency of the effect across the individual experiments [4].

3.8.1 Data Set Preparation

Data from 14 participants (7 pairs) were eliminated: 1 *Explainer-Receiver* pair from OExp as well as REP1, and 5 pairs from REP2. In particular, 5 pairs discussed the design assignment for too short time (less than 2 minutes) and decided to discuss other topics of their interest for the rest of the time. Moreover, the audio quality of the recorded discussion of 2 pairs was bad and the corresponding data from these pairs was eliminated. The final number of participants in each experiment is provided in Table 4.

Table 4: Final number of participants.

ExpID	Initial # of Subjects	Final # of Subjects	Eliminated Pairs	Reason of Elimination
OExp	50	48	1	- Bad quality of recorded discussion
REP1	36	34	1	- Too short discussion time (<2 minutes)
REP2	94	84	5	- Too short discussion time (<2 minutes) (4 cases) - Bad quality of recorded discussion (1 case)
REP3	60	60	0	N/A
Total	240	226	7 pairs	

The discussions between *Explainers* and *Receivers* were recorded by using either mobile phones or *Audacity*, an easy-to-use audio editor and recorder that works on multiple operative systems[13]. We transcribed approximately 23 hours of audio recordings and performed a manual coding of more than 2000 discussion records between *Explainers* and *Receivers*. For coding the discussions, we used the collaborative interpersonal problem-solving skill taxonomy of Margaret McManus and Robert Aiken [38], as presented in Figure 2. This taxonomy captures the collaborative interpersonal communication aspects; Active Discussion, Creative Conflict, and Conversation Management, which we described previously in Section 1. For instance, the following transcribed sentence: "*Can you explain why/how?*" is a *Request* for *Clarification* which contributes to *Active Discussion*. Another example: "*If... then*" refers to *Suppose*; one of the categories of *Argue* which contributes to a *Creative Conflict*. More examples are provided online[14].

NVivo[15] was used for coding the transcriptions. Prior to coding, we ensured coding/rating's reliability by performing two-way mixed Intraclass Correlation Coefficient (I-C-C) tests with 95% confidence interval on 9% of the data. In particular, three coders/rater were involved in measuring the I-C-C of the G group and T group of EXP, REP1, and REP2 (I-C-C value is 0,97 for group G and 0,96 for group T). Whereas, two coders/raters were involved for measuring the I-C-C of the G group and T group of REP3 (I-C-C value is 0,83 for group G and 0,92 for group T). The coding/rating reliability is positive. Indeed, according to [34], I-C-C is good when it is $>$ 0,75 and $\leq$ 0,90 and excellent when it is $>$ 0,90. Based on this result, the raters collaboratively continued to code the rest of the data i.e., 91% of the data.

3.8.2 Descriptive Statistics

By using IBM SPSS[16], we generated descriptive statistics, including *Box-plots* and *Mean +/- 1SD plots*, to analyze the collected data via questionnaires and audio recordings. In particular, we measured: means, medians, standard deviations, and ranges. These descriptive statistics help to analyze central tendencies and dispersion.

[13] `https://www.audacityteam.org`

[14] `http://rodijolak.com/SE_Whispers/Problem_Solving_Skill_Taxonomy.pdf`

[15] `https://www.qsrinternational.com/nvivo`

[16] `https://www.ibm.com/analytics/spss-statistics-software`

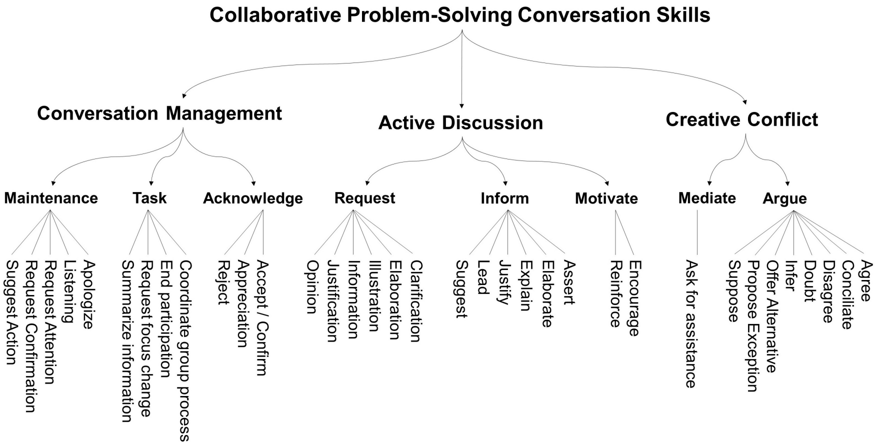


Fig. 2: Collaborative interpersonal problem-solving conversation skills [38].

3.8.3 Hypotheses Testing

In the family of experiments, we wanted to compare two treatments/groups (G and T). So, we assigned our participants to these two groups by following the between-subjects design. In this setting, different people test each condition to reduce learning- and transfer-across-conditions effects. The collected data during the experiments include both interval and ordinal measures. Moreover, they are not normally distributed. Thus, we used non-parametric tests.

In particular, the hypotheses that we formulated in Section 3.7 seek to determine whether two independent samples have the same distribution. Therefore, these hypotheses were tested by performing the non-parametric independent-samples Mann-Whitney test.

3.8.4 Meta-Analysis

We perform a fixed-effect meta-analysis, as all factors that could influence the effect size are the same in all the experiments [4]. We use different scales to measure the communications aspects. Thus, for each experiment (i), we compute the effect size (G_i) by calculating the Hedges' g metric [22]. The assigned weight to each experiment is:

$$W_i = \frac{1}{V_{G_i}} \tag{1}$$

where, V_{G_i} is the within-experiment variance for the ith experiment.

We obtain the global effect size by calculating the weighted mean M:

$$M = \frac{\sum_{i=1}^{k} W_i G_i}{\sum_{i=1}^{k} W_i} \tag{2}$$

According to [22], the effect size is small when $g \geq 0{,}2$; medium when $g \geq 0{,}5$; and large when $g \geq 0{,}8$. We report the result of the meta-analysis by using forest plots [4].

4 Results

In the first part of this section, we report the participants' perceptions of their design experiences and communication skills (the results of the pre-task questionnaire). After that, we present the results of the individual experiments and the performed meta-analysis. Finally, we report the participants' perceptions of their experience in working with different design representations (the results of the post-task questionnaire).

4.1 Perceived Design Experience and Communication Skills

The perceived (based on self-evaluations) design experience and communication skills are detailed here[17]. In summary, we find that:

- the participants are somewhat familiar with software design.
- the participants are familiar with software modeling and good in understanding and sense-making of UML models.
- the participants are very good in reading, understanding, and sense-making of textual documentation.
- the *Explainers* in the group G are neither poor nor good in explaining their knowledge, while the *Explainers* in the group T are good in explaining their knowledge.
- the *Receivers* of the two groups (G and T) are good in building knowledge from conversing with others.

There are no statistically significant differences in the perceived design experience and communication skills between groups G and T in the different experiments. Accordingly, we assume that the design experience and communication skills of participants are not influencing the results of this study.

4.2 Individual Experiments

Table 5 shows the descriptive statistics of the studied communication aspects sorted by two subgroups of studies:

- *Subgroup A*: including OExp, REP1, and REP2.
- *Subgroup B*: including REP3.

Considering *Subgroup A*, we observe that the unbiased estimate of the effect size, based on the standardized mean difference between group G and T (Hedges'g [22]), is positive for *Explaining*, *Recall*, *Active Discussion*, and *Creative Conflict*. This means that there is a clear tendency in favor of using

[17] `http://rodijolak.com/SE_Whispers/PreTask_Results.pdf`

Table 5: Descriptive statistics.

Study ID	**Dept.**	**GSD**			**TSD**			**g***
(Subgroup)	**Var.**	*Mean*	*Median*	*Std. Dev.*	*Mean*	*Median*	*Std. Dev.*	
OExp (A)	EXP	3,864	4,000	0,710	3,654	4,000	0,689	0,295
	UND	1,545	1,750	0,754	1,808	2,000	0,906	-0,307
	REC	5,843	5,535	1,828	5,418	5,205	2,368	0,195
	AD	0,512	0,494	0,094	0,461	0,500	0,137	0,412
	CC	0,267	0,272	0,053	0,234	0,224	0,080	0,466
	CM	0,221	0,209	0,088	0,305	0,244	0,172	-0,580
REP1 (A)	EXP	3,900	4,000	0,553	3,563	4,000	1,209	0,365
	UND	1,875	2,000	0,741	1,625	1,750	0,806	0,317
	REC	6,973	7,165	1,747	5,598	6,125	1,981	0,725
	AD	0,457	0,447	0,107	0,409	0,411	0,063	0,505
	CC	0,166	0,189	0,067	0,118	0,121	0,053	0,745
	CM	0,377	0,406	0,124	0,473	0,488	0,085	-0,841
REP2	EXP	3,810	4,000	0,634	3,714	4,000	0,708	0,140
	UND	1,905	2,000	0,813	1,619	1,500	0,847	0,341
	REC	5,876	5,960	1,685	5,537	5,585	1,787	0,193
	AD	0,488	0,495	0,068	0,431	0,435	0,074	0,786
	CC	0,267	0,250	0,097	0,265	0,249	0,082	0,020
	CM	0,245	0,220	0,086	0,304	0,306	0,056	-0,793
Study ID	**Dept.**	**GSD**			**Altered-TSD**			**g***
(Subgroup)	**Var.**	*Mean*	*Median*	*Std. Dev.*	*Mean*	*Median*	*Std. Dev.*	
REP3 (B)	EXP	4,833	5,000	0,379	3,967	4,000	0,964	1,168
	UND	1,967	2,000	0,798	1,800	2,000	0,726	0,216
	REC	5,948	6,160	1,975	6,664	7,080	2,226	-0,336
	AD	0,571	0,564	0,078	0,625	0,627	0,110	-0,546
	CC	0,184	0,162	0,069	0,113	0,118	0,058	1,086
	CM	0,245	0,232	0,069	0,262	0,255	0,114	-0,179

* Hedges'g: unbiased estimate of the effect size based on the standardized mean difference [22].

GSD over TSD. Regarding *Understanding*, the participants achieved better results when using TSD in OExp (negative g value). In contrast, the participants of REP1 and REP2 achieved better results when using GSD. Regarding *Conversation Management*, the results show that the participants of all the experiments spent more effort on conversation management when using TSD.

Considering *Subgroup B*, we observe that the unbiased estimate of the effect size (i.e., Hedges'g) is positive for *Explaining*, *Understanding*, and *Creative Conflict*. This means that there is a clear tendency in favor of using GSD over Altered-TSD. Regarding *Recall* and *Active Discussion*, the participants achieved better results when using the Altered-TSD. Moreover, the participants spent more effort on conversation management when using Altered-TSD.

We tested whether or not the distribution of the communication aspects (i.e., dependent variables) is the same across the two groups (G and T) by running the Independent-Samples Mann-Whitney U Test.

Table 6 shows the results of the test. The p-value is the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. We set the probability of type I error (i.e., α, probability of finding a significance where there is none) to 0,05. The statistical power is the probability that a test will reject a null hypothesis when it is in fact false. As the power increases, the probability of making a type II error (β-value) decreases. A power value of 0,80 is considered as a standard for adequacy [16]. β-value is used to estimate the probability of accepting the null hypothesis when it is false.

Table 6: Independent variables Mann Whitney Test.

ID	**Subjects #**	**Dependent Variable**	**Mann Whitney Test**		
(Subgroup)	(Sample Size)		*p-value*	*Statistical Power*	*β-value*
OExp (A)	**48** Group G (22) Group T (26)	Explaining	0,367	0,168	0,832
		Understanding	0,300	0,179	0,821
		Recall	0,501	0,101	0,899
		Active Discussion	0,622	0,167	0,833
		Creative Conflict	0,140	0,198	0,802
		Conversation Mgt.	0,284	0,188	0,812
REP1 (A)	**34** Group G (18) Group T (16)	Explaining	0,519	0,184	0,816
		Understanding	0,342	0,151	0,849
		Recall	**0,037**	0,554	0,446
		Active Discussion	0,374	0,184	0,816
		Creative Conflict	0,110	0,550	0,450
		Conversation Mgt.	0,091	0,414	0,586
REP2 (A)	**84** Group G (43) Group T (41)	Explaining	0,636	0,097	0,903
		Understanding	0,152	0,332	0,668
		Recall	0,350	0,139	0,861
		Active Discussion	**0,010**	0,694	0,306
		Creative Conflict	0,990	0,050	0,950
		Conversation Mgt.	**0,011**	0,705	0,295
ID	**Subjects #**	**Dependent Variable**	**Mann Whitney Test**		
(Subgroup)	(Sample Size)		*p-value*	*Statistical Power*	*β-value*
REP3 (B)	**60** Group G (30) Group T (30)	Explaining	**0,000**	0,991	0,009
		Understanding	0,330	0,127	0,873
		Recall	0,115	0,239	0,761
		Active Discussion	0,171	0,291	0,709
		Creative Conflict	**0,017**	0,800	0,200
		Conversation Mgt.	0,740	0,075	0,925

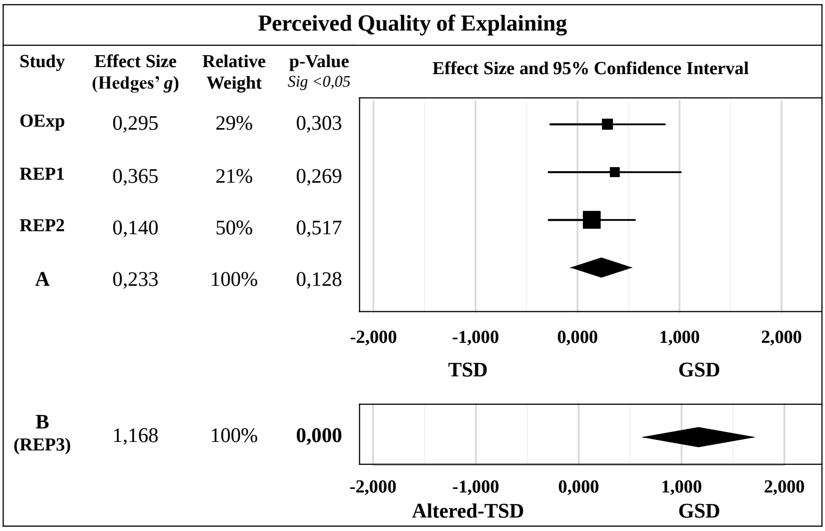
Considering *Subgroup A*, we observe in REP1 that there is a statistically significant difference in *Recall* between the two groups G and T (p-value = 0,037 < 0,05, statistical power is 0,554). In REP2, we observe that there is a statistically significant difference in *Active Discussion* and in *Conversation Management* between the two studied groups (p-values = 0,010 and 0,011 < 0,05, statistical powers are 0,694 and 0,705, respectively).

Considering *Subgroup B*, we observe a statistically significant difference in *Explaining* and *Creative Conflict* between the two studied groups (p-values = 0,000 and 0,017 < 0,05, statistical powers are 0,991 and 0,800, respectively).

4.3 Meta-Analysis

In this section, we report and discuss the meta-analysis by means of forest plots. The squares in each forest plot indicate the effect size of each experiment. The size of the squares represents the relative weight (squares are proportional in size to experiments' relative weight). The horizontal lines on the sides of each square represents the 95% confidence interval. The diamond shows the global effect size (the location of the diamond represents the effect size), while its width reflects the precision of the estimate (i.e., 95% confidence interval). The plot also shows the values of the effect size, weight, and p-value relative to each experiment and to the global measure.

Positive values of the effect size indicate that the use of GSD increases/ improves the effort/quality of the communication aspect, while negative values indicate that using TSD/Altered-TSD is the increasing/improving condition.



Perceived Quality of Explaining

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	0,295	29%	0,303
REP1	0,365	21%	0,269
REP2	0,140	50%	0,517
A	0,233	100%	0,128
B (REP3)	1,168	100%	**0,000**

Fig. 3: Meta-analysis for the perceived quality of Explaining.

4.3.1 Explaining

Figure 3 shows the forest plot for perceived quality of *Explaining* in the two subgroups of studies, A and B. We observe that the effect size values are positive in all the experiments. This implies that using a GSD has a positive effect on perceived Explaining quality. In other words, the participants' level of perceived explaining is better when using the GSD. Despite this tendency, the global effect size of the studies within *Subgroup A* is not statistically significant (p-value is 0,128 > 0,05). In contrast, the global effect size of the study in *Subgroup B* is statistically significant (p-value is 0,000 < 0,05).

Observation 1 (Quality of Explaining).
We find that using a GSD has a positive effect on perceived Explaining *quality. Considering Subgroup A, this effect is not statistically significant. Considering Subgroup B, the effect is statistically significant. This suggests that the perceived quality of design explaining is better when using a GSD than Altered-TSD.*

4.3.2 Understanding and Recall

In a revised Bloom's taxonomy, Anderson et al. [1] outline a hierarchy of cognitive-learning levels ranging from remembering of a specific topic, over understanding and application of such knowledge, to advanced levels of analysis, evaluation, and creation. Figure 4 shows the hierarchy of the six cognitive learning levels. According to Anderson, *remember* is the recalling of the previously learned topic. *understand* is the ability to grasp the meaning of the topic

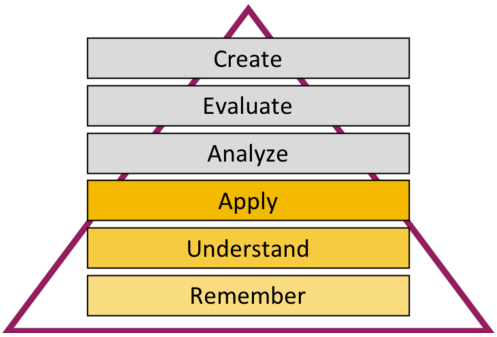


Fig. 4: Bloom's taxonomy of cognitive learning.

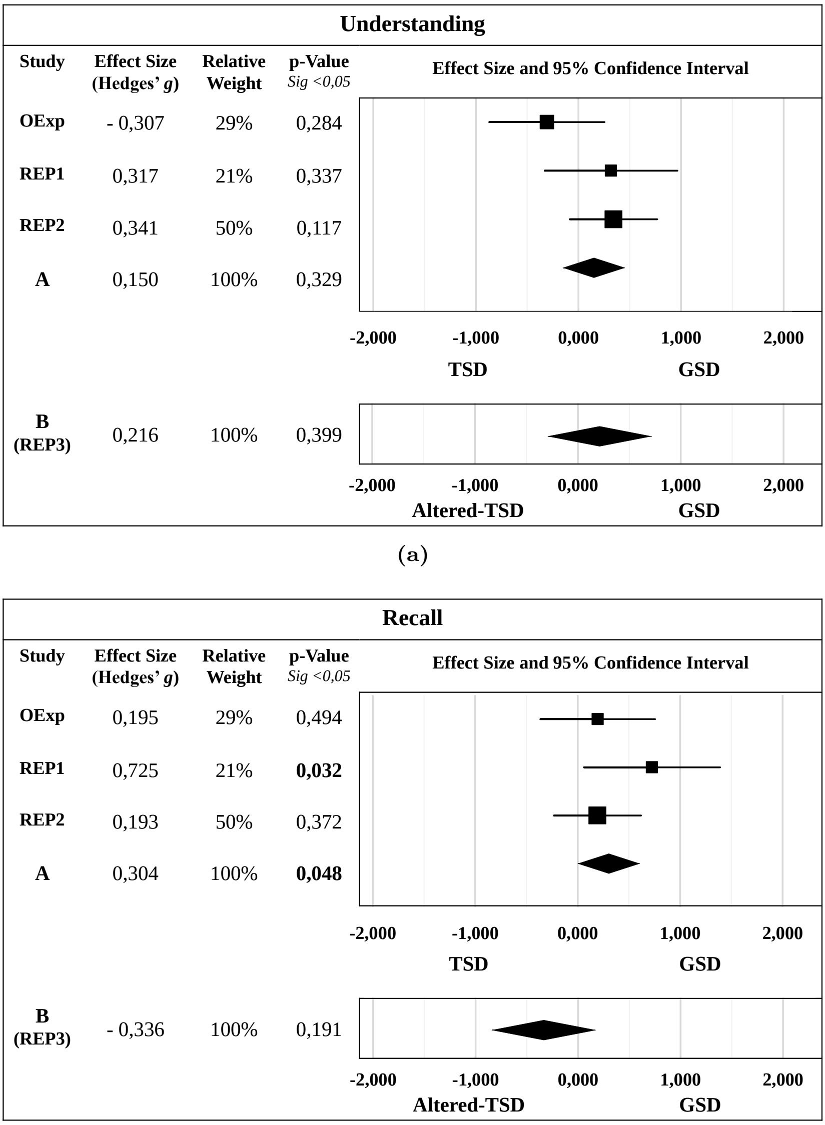
by interpreting the knowledge and predicting future trends. *Apply* instead, comes on top of *understand*. It is the ability to use the acquired and comprehended knowledge in a new and concrete context or situation. In order to measure the quality of understanding of our experiments' participants, we formulated three questions on design *maintenance* (these questions are provided in Section 3.6) which required the participants to use/apply their acquired knowledge in a new context (i.e., *apply* in Anderson's revised taxonomy).

The participants in the two groups (G and T) answered ten recall questions. We formulated the recall questions (see Section 3.6) to evaluate how well the participants do remember the design details after the discussions.

Figure 5 shows the forest plot for quality of (a) *Understanding* and (b) *Recall* ability of design details. Regarding the quality of *Understanding*, the effect size value is negative for OExp, which means that TSD is the improving condition. For the other experiments in *subgroups A* and *B* the values of the effect size are positive. This implies that using a GSD in these experiments has a positive effect on the understanding quality. Despite these tendencies, the global effect size of *subgroups A* and *B* is not statistically significant (p-values are 0,329 and 0,399, respectively). Considering the *Recall* ability, we observe that the effect size values in *subgroup A* are positive. This implies that using a GSD has a positive effect on the *Recall* ability. This effect is statistically significant and has a medium effect size for REP1. Furthermore, the global effect size is statistically significant (p-value $= 0{,}048 < 0{,}05$). In contrast, the effect size value in *subgroup B* is negative. This implies that using a Altered-TSD is the improving condition. However, the effect is not statistically significant (p-value $= 0{,}191 > 0{,}05$).

Observation 2 (Quality of Understanding).
In OExp, we find that using a TSD has an advantage in improving Understanding*. Whereas in the other experiments (REP1, REP2, and REP3), we find that using a GSD is the improving condition. Globally, there is no statistically significant difference in the quality of understanding between the two groups: G and T.*

Observation 3 (Recall Ability).
Considering the experiments of Subgroup A, we find that using a GSD has a positive, statistically significant effect on Recall *ability. This suggests that using a GSD during design communication has an advantage over TSD in improving the recall of design details. In REP3, we find that using a Altered-TSD has an advantage in improving the* Recall *ability. However, the effect is not statistically significant.*



Understanding

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	- 0,307	29%	0,284
REP1	0,317	21%	0,337
REP2	0,341	50%	0,117
A	0,150	100%	0,329
B (REP3)	0,216	100%	0,399

(a)

Recall

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	0,195	29%	0,494
REP1	0,725	21%	**0,032**
REP2	0,193	50%	0,372
A	0,304	100%	**0,048**
B (REP3)	- 0,336	100%	0,191

(b)

Fig. 5: Meta-analysis for quality of Understanding (a) and Recall ability (b).

4.3.3 Interpersonal Communication

Figure 6 shows the the forest plot for collaborative interpersonal communication dimensions: *Active Discussion* (AD), *Creative Conflict* (CC), and *Conversation Management* (CM).

Considering AD, we observe that the effect size values of *Subgroup A* studies are positive. This implies that using a GSD has a positive effect on the amount of ADs. The global effect size for AD is statistically significant (p-value = 0,005 < 0,05). The effect size value of *Subgroup B* study is negative. This implies that using a Altered-TSD is the improving condition. However, the effect size of the study in *Subgroup B* is not statistically significant (p-value = 0,131).

Considering CC, we observe that the effect size values of all the studies are positive. This implies that using a GSD has a positive effect on the amount of CCs. The global effect size of *Subgroup A* is not statistically significant (p-value = 0,162 > 0,05). In contrast, the effect size of *Subgroup B* study is statistically significant (p-value = 0,004 < 0,05).

Considering CM, we observe that the effect size values of all the studies are negative. This implies that the effort on CM is bigger when using TSD/Altered-TSD. The global effect size of *Subgroup A* is medium and statistically significant (p-value = 0,001 < 0,05). The effect size of the study in *Subgroup B* is not statistically significant (p-value = 0,615 > 0,05).

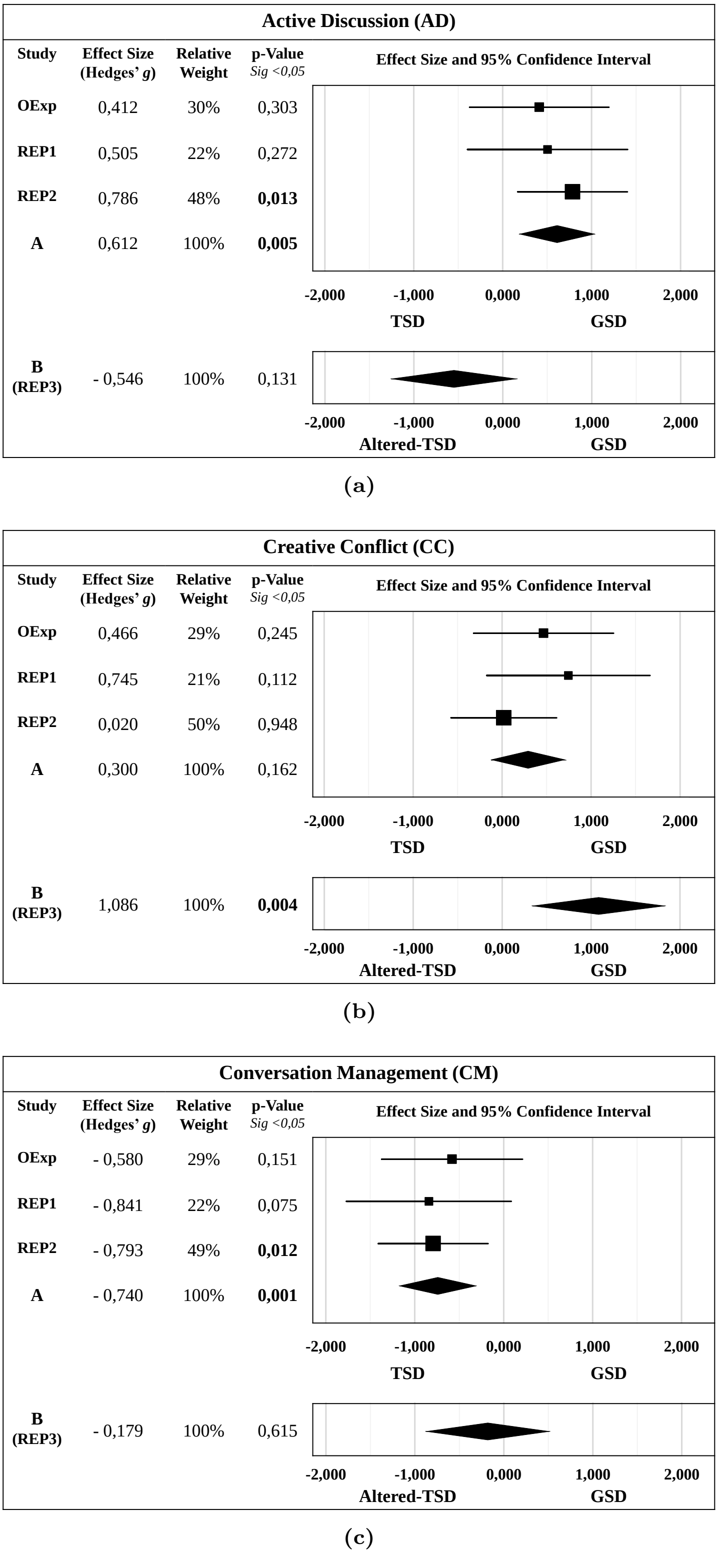
Observation 4 (Active Discussion, AD).
We find that a GSD fosters more AD than a TSD. The effect is statistically significant. This suggests that GSD' users question, inform, and motivate each other more than TSD' users.

Observation 5 (Creative Conflict, CC).
We find that using a GSD has a positive effect on the amount of CC discussions. This effect is not statistically significant in Subgroup A. In Subgroup B, this effect is statistically significant. This suggests that GSD' users argue and reason about others' discussions more than Altered-TSD' users.

Observation 6 (Conversation Management, CM).
A GSD requires less CM effort than TSD/Altered-TSD. The effect is statistically significant for Subgroup A. *This suggests that GSD' users do less coordination and acknowledgment of communicated information than TSD' users.*

4.4 Motivated and Cohesive TSD

Falessi et al. [17] suggested that documentation of software design rationale could support many software development activities, including analysis and re-design. Tang et al. [56] conducted a survey of practitioners to probe their perception of the value of software design rationale and how they use and document it. They found that practitioners recognize the importance of document-



Active Discussion (AD)

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	0,412	30%	0,303
REP1	0,505	22%	0,272
REP2	0,786	48%	**0,013**
A	0,612	100%	**0,005**
B (REP3)	- 0,546	100%	0,131

(a)

Creative Conflict (CC)

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	0,466	29%	0,245
REP1	0,745	21%	0,112
REP2	0,020	50%	0,948
A	0,300	100%	0,162
B (REP3)	1,086	100%	**0,004**

(b)

Conversation Management (CM)

Study	Effect Size (Hedges' *g*)	Relative Weight	p-Value *Sig <0,05*
OExp	- 0,580	29%	0,151
REP1	- 0,841	22%	0,075
REP2	- 0,793	49%	**0,012**
A	- 0,740	100%	**0,001**
B (REP3)	- 0,179	100%	0,615

(c)

Fig. 6: Meta-analysis for collaborative interpersonal communication

ing design rationale for reasoning about their design choices and supporting the subsequent implementation and maintenance of systems.

The goal of running REP3 is to know how a *motivated* and *cohesive* TSD (as described previously in Section 3.5.1 – Altered TSD) could influence design communications. To this end, we used a different TSD in REP3, which includes a rationale that motivates why the MVC paradigm is selected for structuring the design. Moreover, we organized the information/knowledge in the TSD and made it cohesive. In particular, the relationships of each entity are reported right after describing it, instead of being reported with all the other relationships in the 'relationship section' at the end of the TSD.

To achieve the goal of REP3, we use a fixed-effect subgroup analysis [4] to determine whether the Altered-TSD variant is more effective than the TSD. In particular, we compare the mean effect for two subgroups of studies:

- *Subgroup A*: the experiments that use a TSD (OExp, REP1, and REP2).
- *Subgroup B*: the experiment that uses a Altered-TSD (REP3).

For each subgroup of studies, we report in Table 7 the mean effect size and variance of the studied communication aspects. We observe that effect size of REC and AD is lower in *Subgroup B*. This indicates that the Altered-TSD is better than TSD in promoting *Recall Ability* and *Active Discussion*. We also observe that the effect size of CM is higher in *Subgroup B*. This indicates that the users of Altered-TSD spent less effort on *Conversation Management*.

To statistically analyze the difference between TSD and Altered-TSD, we use the Z-test method [4]. The results of the test are presented in Table 8. We observe three significant differences at the α level of 0,05 concerning *Explaining* (EXP), *Recall* (REC), and *Active Discussion* (AD).

For EXP, we find that the two-tailed p-value corresponding to Z = 2,290 is 0,003. This tells us that the TSD of *Subgroup A*' studies is better for *Explaining* than the Altered-TSD of *Subgroup B*'s study. In addition to reporting the test of significance, we report the clinical significance. The difference in effect size between the two subgroups of studies is *Diff*= 0,935, and the 95% confidence interval is in the range of 0,190 and 0,715.

For REC and AD, we find that the two-tailed p-value corresponding to Z = 2,136 and 2,740 are 0,033 and 0,006, respectively. This tells us that the Altered-TSD of the *Subgroup B*'s study is better than the TSD of *Subgroup A*' studies for increasing *Active Discussions* and enhancing *Recall* ability of design details. The differences in REC and AD effect size between the two subgroups of studies are *Diff*= 0,639 and 1,158, and the 95% confidence intervals are in the range of -0,124 and 0,393 for REC, and in the range of -0,062 and 0,670 for AD.

Observation 7 (Cohesive and Motivated TSD)
Based on empirical findings, we find that a cohesive *TSD that* motivates *the design choices with rationale increases the amount of* active discussions *and enhances the* recall *ability of design details at the cost of reducing the perceived quality of* explaining.

Table 7: Effect size and variance of communication aspects of Subgroup A and Subgroup B

Dept. Var.	**Subgroup A**		**Subgroup B**	
	Effect size (Hedges' g)	*Variance*	*Effect size (Hedges' g)*	*Variance*
EXP	0,233	0,023	1,168	0,076
UND	0,150	0,024	0,216	0,065
REC	0,304	0,024	-0,336	0,066
AD	0,612	0,047	-0,546	0,131
CC	0,300	0,046	1,086	0,146
CM	-0,740	0,048	-0,179	0,127

Table 8: Subgroup analysis of the difference between TSD and Altered-TSD

Dept. Var.	**Statistical Significance**		**Clinical Significance**		
	Z-test	*p-value (2-tailed)*	*Abs. ES. Diff.*	*Lower CL*	*Upper CL*
EXP	2,960	**0,003**	0,935	0,190	0,715
UND	0,220	0,826	0,066	-0,091	0,425
REC	2,136	**0,033**	0,639	-0,124	0,393
AD	2,740	**0,006**	1,158	-0,062	0,670
CC	1,794	0,073	0,786	0,122	0,856
CM	1,341	0,180	0,561	-0,952	-0,219

Abs. ES. Diff: Absolute Effect Size Difference
C.L.: Confidence Level

4.5 Perceived Experience in Working with Different Design Representations

The perceived experience (based on self-evaluations) in working with different design representations are detailed here[18]. In summary, we find that:

- the participants (both *Explainers* and *Receivers*) perceive that the design of the system is very clear.
- the participants perceive that they are good in recalling a conversation.
- the participants in group G perceive that they are good in recalling UML design models.
- the participants in group T perceive that they are good in recalling a textual documentation.
- the *Explainers* in group G perceive that models are helpful in understanding the design.
- the *Explainers* in group T perceive that having diagrams would have helped in understanding the design.
- the *Explainers* in group G perceive that models are very helpful in explaining the design.
- the *Explainers* in group T perceive that having diagrams would have helped in understanding the design.
- the *Receivers* in group G perceive that models are helpful in understanding the design.
- the *Receivers* in group T perceive that having diagrams would have helped in understanding the design.

[18] `http://rodijolak.com/SE_Whispers/PostTask_Results.pdf`

5 Discussion

Our experiments investigate whether design communication between software engineers can become more effective when using GSD instead of TSD to exchange design information. To this end, we investigate whether using a GSD affects six considered communication aspects (Understanding, Explaining, Recall, Active Discussion, Creative Conflict, and Conversation Management) differently from using a TSD (R.Q.1). Moreover, we study whether a *cohesive* and *motivated* TSD (i.e., Altered-TSD) improves design communication (R.Q.2).

Considering *Subgroup A*, the global effect size of the perceived explaining quality is positive. This means that using a GSD has a positive effect on the perceived explaining quality. Similarly, the global effect size of the understanding (i.e., maintenance task) score is positive, which means that the score of the GSD users is better than the score of TSD users. Nevertheless, by considering distributions of the scores we neither find a statistically significant difference in the quality of explaining (Observation 1) nor in the quality of understanding (Observation 2) between the two groups: G and T.

While analyzing the recorded, and further transcribed, discussions between the *Explainers* and *Receivers*, we interestingly observed a difference in the explaining approach between the *Explainers* of the two groups. Figure 7 provides an illustration of the observed explaining approaches in the two groups. On the one hand, the *Explainers* of a TSD tended to explain the three modules of the MVC sequentially: Firstly the model entities, then the controllers, and lastly the views, as these modules are orderly presented in the textual document. We think that this trend is intrinsically imposed by the nature of textual descriptions where the knowledge is presented sequentially on a number of consecutive ordered pages. On the other hand, the *Explainers* of the GSD had more freedom in explaining the design. Indeed according to their explaining preferences, the *Explainers* of the GSD tended to jump back and forth between the three MVC modules when explaining the design. Based on this, we suggest that a GSD has an advantage over the TSD in unleashing *Explainers*' expressiveness when explaining the design, as well as in helping navigation and getting a better overview of the design. However, developers might not have this advantage when explaining many GSDs (e.g., many UML diagrams) spread on different pages within a software design documentation.

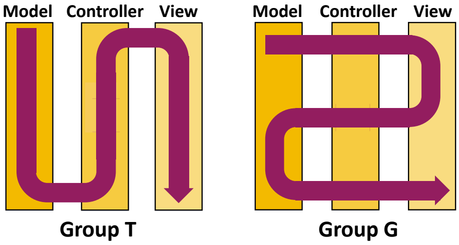


Fig. 7: Observed explaining approaches used in the two groups: G and T.

We found that using a GSD is better than a TSD for recalling the details of the discussed design (Observation 3). This is actually inline with Maede et al. [39], who suggest that drawing graphical notations brings more recall benefits than writing textual words in younger and older adults.

Graphical representations are considered better than the textual in representing information which deals with relationships between entities [60]. One of the recall questions that we used to measure the recall ability of the participants is concerned with the relationships between the entities of the software architecture design. We compared the score (interval variable; min is 0 and max is 1 point) of the two groups on this question. On average, the users of the graphical representation were slightly better in recalling the relationships between the entities (G: Mean= 0,506; Std. Dev.= 0,331) vs. (T: Mean= 0,423; Std. Dev.= 0,347). However, this difference is not statistically significant (Sig.= 0,128 > 0,05; Hedges' g= 0,244; Power= 0,338).

The *Chinese Whispers* game is often invoked as a metaphor for miscommunication. In this game, the first player often fails to recall all the information of the initial message that she/he receives. Likewise, the second player often fails to recall all the information of the message that she/he receives from the first player, and so on for the rest of the players. In the same manner, the *Explainers* in our experiments failed to *recall* all the design details that we asked for in the post-task questionnaire (Mean Score= 3,319; Std. Dev.= 0,855). The Receivers were, as expected, worse than the *Explainers* in recalling the design details (Mean Score= 2,492; Std. Dev.= 0,885). Moreover, we found that the difference in *recall* ability betweeen *Explainers* and *Receivers* is statistically significant (Sig.= 0,000 < 0,05; Hedges' d= 0,946, Power= 0,999).

Based on empirical results, we find that a GSD fosters more *Active Discussion* (AD) than TSD (Observation 4), while reducing *Conversation Management* (CM) at the same time (Observation 6). In the skill taxonomy of McManus and Aiken [38], the communication activities in the AD category generally aim at helping an active exploration of the discussed argument by encouraging information requesting, clarification, or elaboration. In contrast, the branch of CM comprises communication activities that generally contribute less to active information requesting or clarification, such as acknowledging or coordinating group tasks. Consequently, we suggest that using a GSD as a basis for software design communication promotes an active exploration of the communicated designs, which in turn helps to improve the effectiveness of software design communication.

There is no significant difference in Creative Conflict (CC) discussions between group G using GSD and group T using TSD (Observation 5). We suggest that the type of design description does not influence design argumentation and reasoning. Alternatively, we think that the context, complexity of the design, available knowledge, or the application of reasoning techniques might affect the quality of design argumentation and reasoning discussions, as suggested by Tang et al. [57].

It is widely assumed that model-based techniques support communicating software [26]. Our findings support such assumption and prove that using a

GSD improves the recall ability of the discussed design details, fosters *Active Discussion*, and at the same time reduces less useful conversation on activities management.

We conducted REP3 to better calibrate our findings of the differences between GSD and TSD. We found that a motivated (i.e., augmented with rationale) and cohesive TSD helps to enhances the *recall* of the design details and increases the amount of *active discussions* at the cost of reducing the perceived quality of *explaining* (Observation 7). This finding is indeed inline with Tang et al. [55] who stated that discussing the reasons of making software design choices (i.e. design rationale) positively contributes to the effectiveness of software design discussions by facilitating communication and design knowledge transfer. However, we found that adding more details (e.g., rationale) to the TSD adversely influences the perceived quality of *explaining.* One explanation for this effect is that the *Explainers* did not have enough time to explain the details of the Altered-TSD. For the same reason, the *Receivers* might have perceived that the *Explainers* did not go through the entire textual description when explaining the software design.

5.1 Threats to Validity

Our family of experiments is subject to threats to their construct validity, internal validity (causality), external validity (generalizability), and conclusion validity. We highlight these issues and discuss related study design decisions.

5.1.1 Construct Validity

Constructs validity refers to how well operational measures represent what researchers intended them to represent in the study in question. In this study, we used a single method for measuring the impact of different design representation per each communication aspect. To mitigate this issue, we did not only rely on questionnaires, but also recorded, transcribed, and later evaluated the communication observed during the experiments. Nonetheless, leveraging additional methods to probe the explaining, understanding, recall, and interpersonal communication skills of the participants might help to better investigate the effects of different design representations. Such methods, for instance, might comprise conducting actual software design or software engineering tasks after receiving the explanation. However, this would introduce a multitude of other variables (e.g., the programming language or IDE used) that either can be hardly controlled or demand for drastic simplification, thus reducing our experiments' generalizability.

Another threat to construct validity could arise from discretizing the measurement of continuous properties, such as the participants' familiarity with software design or their expertise with UML. This challenge has been investigated for balanced *Likert* and identified as not compromising generalizability [45].

5.1.2 Internal Validity

The questionnaires to evaluate the participants' performance raise threats to internal validity themselves: For instance, the participants might interpret the Likert scales we have used differently, might have avoided extreme responses (central tendency bias), and - as the participants evaluated their communication skills themselves - might be biased towards overestimating or underestimating their skills, which might be subject to different effects on their introspection. To support comprehension and reproduction of results, we use established surveys where possible and provide all materials on the experiments' companion website. Nonetheless, completely mitigating the potential effects of surveys' general deficiency requires the development of novel methods to test familiarity and understanding of UML designs and textual designs, as well as communication skills. While for the latter, specifically tailored exercises might be feasible to evaluate the skill level, conducting these, (a) requires unbiased instruments as well and (b) might affect our experiments. A specific challenge of our family of experiments regarding the questionnaires arises from conducting the REP2 survey in French, whereas the other experiments used English documents. While this generally could affect the results, the experimenters of REP2 had the task documents and questionnaires professionally translated and reviewed to maintain the consistency of the communicated information.

To mitigate the effect of limited preparation and explanation time – the *Explainers* had 20 minutes to understand the design and 12 to discuss it with the *Receivers* – we conducted multiple pilot studies at all sites prior to the actual experiments to understand how much time is required. After running the pilot studies, we increased the initially considered 10 minutes of discussion to 12 based on the feedback of the participants of the pilot studies. Afterwards, we conducted another pilot study that confirmed that both times are considered suitable for the tasks.

Other challenges to internal validity stem from the selection of our experiments' participants. Potential confounding factors include that due to randomly assigning the participants to the G or T group, certain personality types are prevalent in one of the groups – which could affect results. By measuring the Big Five factors of personality [14], we checked that this is not the case: the distribution of the five personality factors (Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness) is the same across the two groups. Similarly, it could have affected our findings that the members of one of the two groups have significantly more experience with software design than the members of the other group. The pre-task questionnaire establishes that this is not a problem of our study. Other issues could have arisen from our participants being unfamiliar with UML designs or textual designs but the pre-task questionnaire shows that this is not the case. We assume that this is due to the participants' educational backgrounds (in which processing textual designs for exercises or exams is common).

The textual representations used in this research are structured by indentation, indexing, and grouping information, which are helpful for information retrieval [8]. However, these might have positively affected the quality of TSD communication. Similarly, the MVC entities in the graphical representation were highlighted by colors, which is also helpful for information retrieval [8]. This might have also positively affected the quality of GSD communication. If the descriptions of the entities in TSD were tangled and if the entities of the GSD were not colored, then the quality of communication of these two representations might have been different and less efficient. As we use different enhancement techniques for GSD and TSD, it is possible that this affected the results of the comparison. Indeed, the augmentations to the textual representation might yield other (stronger or weaker) effects than the class diagram coloring. As both, coloring in graphical models and structuring of textual design documents, is common in industrial practice, we do not consider this a significant threat over using unstructured text and uncolored diagrams.

Some *Receivers* of the text group were drawing (informal) class diagrams while being explained to. Hence, there might be an interaction of both treatments, but with only six (2,5%) of the *Receivers* being affected, the effect of this combination of both representations is negligible.

Another threat might arise from using textual survey questions as the method to investigate the benefits of textual and graphical designs. Maybe, textual design representations yielded better answers to the questions because they are syntactically closer than graphical designs to the textual answers. This threat could be mitigated through leveraging graphical questions and answers in the surveys. While this would be feasible for the answers, for formulating the questions as graphical class diagrams, this would entail a new syntax which might yield further threats.

5.1.3 External Validity

Threats to external validity indicate to which extent the results of our study can be generalized. Due to working at software engineering research and education institutes, we selected students with strong software engineering backgrounds of our Universities. While this prevents generalizing results to software developers with different backgrounds (e.g., developers in computer vision, artificial intelligence, or robotics), software design aims at software architecture from which we expect strong software engineering backgrounds.

Also, we conducted our studies with students instead of software design practitioners. Hence, the participants involved in our experiments may not represent the general professional population of software engineering practitioners. While this limits us from generalizing our findings to other subjects (i.e., domain experts, professional software architects, industrial practitioners in the field), the differences between students and professional software developers in performing small tasks are generally very small [25]. We, therefore, consider our findings as a basis to extend our study to a larger community of software engineering practitioners.

Another threatening effect is that the population of professional software developers yield a larger age range than students. With recall abilities changing over time [9], this limits generalization of our results to professional software developers of the same age range – between 20 and 30 years – than software engineering students and PhD students (as proposed in [18]).

Moreover, the studies were conducted in educational contexts, i.e., contexts in which the students usually are evaluated and graded. This generally might have improved their performance (Hawthorne effect). However, as this applies to both groups, this does not affect our results.

Due to the outline of our experiments as single one-hour sessions and their popular context in sports that are easily relatable, we can exclude threats regarding history or maturation. The participants could neither have been effected from previous events of the experiment as there have not been any.

Moreover, as we used the same two textual/graphical notations in all experiments, this limits generalizability of our results to other textual or graphical representations, i.e., differently structured text or differently highlighted class diagrams. This, however, is a threat independent of the specific choice of representation and demands for studies deploying multiple (popular) representations – which demands correctly identifying industrially relevant forms of representation and yields further threats to generalizability.

The use of a single case specification is a threat to the generalizability of the results. The size, topic, and complexity of the design case specification might affect the communication quality and the results of the comparison. This threat can be addressed by conducting replication studies with different design case specifications.

Another challenge to generalizability might arise from the constructs investigated, i.e., whether structured textual design documents and colored UML class diagrams actually are relevant to communicating design decisions in industry. While the use of UML in software design and engineering is undaunted in various domains (cf. [36,64]), so is the use of textual documents to describe software designs [7,61,42]. However, using a specific form of structured text for communicating design decisions limits generalizability to this form of text. For instance, in requirements engineering, there are different tools that support capturing textual requirements and design decisions using different textual representations [6] and using these might entail different effects.

Generalizability might also be challenged by the size of documents used of investigation. There are no studies on the number of classes per class diagram in industrial software engineering projects. However, a report on numbers of classes per class diagram used in different lectures reports that in 101 diagrams from 5 different courses, the maximum number of classes per diagram is 40, with the minimum being 3 and the average being 10.75 [63]. This might indicate that our design class diagram of 28 classes is a bit more complex than it would be usual for education (and hence be more realistic regarding industrial challenges). Another study investigated 100 android applications from open-source repositories [51]. Here, only the average size of these applications as 90 classes is reported. While this does not report how these would be aligned in different class diagrams, assuming these cover at least three different concerns (e.g., model, view, and controller) appears reasonable, which would entail 30 classes per class diagram on average and would be in line with the 28 classes presented in our experiment. Therefore, we consider the size of the experiments' class diagrams relevant. For the textual design documents, we are unaware of any studies on their average size, but due to them containing the same information as the class diagrams, which are of relevant size, we conclude that these should be as well. However, this needs further investigation and might challenge the generalizability of our results. Also, the effect of the number of classes conveyed in both representations might affect understanding and recall. This also demands for further investigation.

Similar to the threat of using specifically indented and colored documents, the optimality of their representations might challenge generalizability of our results as it might be conceivable that there are better suitable textual or graphical representations that lead to different results. To the best of our knowledge, the best representations of textual design documents and graphical class diagrams still have to be identified and whether these are optimal for any domain needs to be investigated. Nonetheless, differently presented textual or graphical designs might have yielded different effects. This, however, is a threat to generalizability that holds for any study investigating a finite number of alternative treatments where infinitely many are possible and needs to be considered when applying our results.

Also, the experimental conditions (scope, team size, duration, etc.,) might differ from real-world conditions and limit generalizability of results. Nonetheless, especially in the use case of onboarding of job newcomers by experienced developers and designers, this challenge is of practical interest as indicated by Ericsson's "*Experience Engine*" initiative (cf. section 1).

5.1.4 Conclusion Validity

Threats to conclusion validity challenge how reasonable a research or experimental conclusion is. In our study, these threats might arise, mainly, through concluding the existence of in-existing differences (type I error) and concluding the in-existence of existing differences (type II error).

We conducted hypotheses testing to determine whether two independent variables have the same distribution. We might have committed type I error and incorrectly rejected the null hypothesis (false positive), or committed type II error and incorrectly accepted the null hypothesis (false negative). However, we considered the *significance* and minimized the risk of detecting a non-real effect by setting the α value to 0,05. Also, we analyzed the *sensitivity* by discussing the effect size and statistical power of our tests.

We underline that a small sample size of experiments yields low statistical power which, in turn, increases the likelihood of making type II error (accepting the null hypothesis when it is false). To mitigate this threat, we conducted a family of experiments that aims at maximizing the sample size with repeated measures and increasing the statistical power and precision of the results [49].

5.2 Implications

Using GSD to communicate software designs produces *more active discussion, less conversation management, and better recall.* These effects contribute to deepening the active exploration of the discussed design [21], which is why we consider using GSD beneficial to communicating software designs. For identification of design errors, textual descriptions seem to be more efficient [40] than GSD. Our findings suggest the use of GSD as a basis for communicating designs with the objective of transferring design knowledge, which is in line with the observed benefits of graphical documents on recall [39].

Our findings, however, assume that the textual design document accurately represents the GSD. Often, however, these natural language documents yield ambiguities or omit details that can be missed less easily in graphical descriptions. We assume that this can be due to graphical descriptions, such as UML class diagrams, being accessible for model checking to identify, e.g., missing associations or missing types. Future work should investigate whether textual artifacts used in practice indeed represent the underlying design accurately.

With REP3, we investigated the effects of a cohesive and motivated TSD on design understanding, explaining, recall, and interpersonal communication. As we found a difference in explaining, recalling, and active discussion between both groups: TSD and altered-TSD (Observation 7), future research in improving software design communication should investigate comparing benefits of augmenting GSD with textual motivation and rationale as well.

5.3 Generalization

Generally, we found that communicating design with a GSD yields better discussions and better recall. We believe that these effects are not limited to software design documents but transfer to graphical software descriptions in general. While, for instance, UML class diagrams meant for implementation might differ in the level of detail, but not in the general representation. Applying our findings regarding the benefits of (i) GSD over TSD and (ii) cohesive TSD with rationale to other kinds of software artifacts can yield benefits for their communication and consumption as well. For instance, as requirements documents become more complex [19], augmenting these with graphical representations or rational could, ultimately, improve requirements engineering. Model-based systems engineering [44] traditionally considers graphical representations. Nonetheless, similar improvements could be achieved as the collaborating stakeholders from various domains could benefit from being provided rationale of design decisions made in other domains.

There also is research in textual modeling [24], which leverages textual models with well-defined semantics for software design and development. As such, these textual models are in-between GSD and TSD and whether our results translate to textual software models, such as UML/P class diagrams [47], needs further investigation.

Similarly, the observed benefits of GSD are subject to the viewpoint we selected in a fashion that allows presenting the complete design description (i.e., model) on a single sheet of paper. For more complex diagrams, this might not scale-up. However, we assume that the textual design document (currently three sheets of paper) scales-up even worse. Consequently, we believe that the effects of software design representation on large designs with hundreds or thousands of elements will be even more prominent.

6 Conclusion and Future Work

We conducted a family of experiments to study the effect of using graphical versus textual software design descriptions on software design communication. According to [1,12,53], we considered the following communication aspects:

- *Explaining*: communicating intellectual capital from one person to others.
- *Understanding*: receiving others' intellectual capital.
- *Recall*: recognizing or recalling knowledge from memory to produce or retrieve previously learned information.
- Collaborative Interpersonal Communication, which includes:
  - *Active Discussion*: questioning, informing, and motivating others.
  - *Creative Conflict*: arguing and reasoning about others discussions.
  - *Conversation Management*: coordinating and acknowledging communicated information.

Based on empirical findings, we suggest that a *graphical* software design descriptions (GSD) improves design-knowledge transfer and communication by:

- promoting *Active Discussion* between developers,
- reducing *Conversation Management* effort, and
- improving the *Recall* ability of design details.

Furthermore, compared to its unimproved version, a *well-organized* and *motivated* textual software design description (that is used for the same amount of time) enhances the *recall* of design details and increases the amount of *active discussions* at the cost of reducing the perceived quality of *explaining.*

6.1 Impacts on Practitioners

In a field study of the Software Design process, Curtis et al. [11] identified broad communication and knowledge sharing as two factors that have effects on software quality and productivity. According to our findings, we suggest that the use of GSD can help in improving design-knowledge sharing and communication. Hence, we identify the following impacts on practitioners:

- *Agile Practices.* Agile development practices include several processes in which communication is at least involved, if not central [43]. Daily meetings are, by definition, the perfect example of agile ceremony which completely relies on communication. According to Karlström et al. [32], holding daily meetings as a mechanism for design problem solving appeared to have positive effects on the communication of the design issues. Based on our findings, introducing GSDs in daily discussions about design decisions would enhance the communication quality between participants, which in turn could strengthen the impact of applying agile practices in software engineering projects.

- *Reducing Development Efforts.* Multiple studies demonstrated that communication is one of the most time-consuming tasks in software development, requiring more effort than any other development activity [30] and taking up to two hours a day per each individual developer[65]. As face-to-face communications are strongly preferred when possible [54,65], the use of GSD as a support for design-related communication could be of benefit for productivity. Minimizing the required effort for communication would provide developers with more time at disposal as well as reduce developers mental-load, so they can focus on different tasks.

- *Satisfaction and Productivity.* Although there is no notable difference in the perceived quality of explaining between group G and group T, all participants from the two groups reported that a GSD indeed helped, or would have helped them, in explaining the design. Accordingly, we think that using GSDs would make the communication of the design easier and increase the satisfaction of developers. Graziotin et al. [20] reported that satisfaction is directly correlated to productivity. So, we suggest the use of GSD in design meetings in order to increase the productivity of software development teams.

- *Pedagogical medium.* By observing of the explaining approaches in the two groups, we suggested that a GSD has an advantage over the TSD in helping navigation and getting a better overview of the design. Even though this requires more investigation, we suggest that, due to its nature, a GSD provides more adaptability and extra degrees of *explaining* freedom, which makes it a better pedagogical medium for face-to-face design knowledge transfer.

- *Design Rationale.* Falessi et al. [17] state that documenting design rationale could support many software development activities, such as an impact analysis or major redesign. Tang et al. [58] find that design reasoning (i.e., discussing rationale) improves the quality of software design. In this paper, we find that a TSD that *motivates* the design choices with rationale can enhance the *recall* and *explaining* of its design details. Accordingly, we suggest the producers of software design tools (graphical or textual) to provide explicit mechanisms for capturing and retrieving design rationale. Furthermore, we encourage developers to include design rationale in design documentations to improve design communication, which in turn should improve the overall communication and collaboration, and thus the productivity, in SE projects.

6.2 Future Work

One future direction is to replicate the experiment in order to address and minimize the threats to the validity of our research design and results. For instance, by replicating the experiment with a more complex graphical or textual software design description, by changing the order of complexity of the recall and maintenance tasks, or by involving professionals. Moreover, to maximize the benefits, another line of research is to investigate new techniques or approaches that would enhance the effectiveness of software design communication. One example of these approaches is proposed in a study by Tang et al. [57] where a reminder card approach was employed to improve software design reasoning discussions. Another example is proposed by Robillard et al. [46] who argue that automatic on-demand documentation generators would effectively support the information needs of developers.

Acknowledgements We would like to thank Prof. Robert Feldt for his valuable suggestions and inspiring discussions about this work. Moreover, this work was partially supported by the Scientific Grant Agency of Slovak Republic (VEGA) under the grant No. VG 1/0759/19 and it is partial result of the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

Ethical Issues In this study, we considered the major ethical issues according to [52]: informed consent, beneficence– do not harm, and respect for anonymity and confidentiality.

References

1. Anderson, L.W., Krathwohl, D.R., Airasian, P.W., Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., Raths, J., Wittrock, M.C.: A taxonomy for learning, teaching, and assessing: A revision of bloom's taxonomy of educational objectives, abridged edition. White Plains, NY: Longman (2001)
2. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Transactions on Software Engineering **25**(4), 456–473 (1999)
3. Bobek, E., Tversky, B.: Creating visual explanations improves learning. Cognitive Research: Principles and Implications **1**(1), 27 (2016)
4. Borenstein, M., Hedges, L.V., Higgins, J.P., Rothstein, H.R.: Introduction to meta-analysis. John Wiley & Sons (2011)
5. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice. Synthesis Lectures on Software Engineering **1**(1), 1–182 (2012)
6. Cant, T., McCarthy, J., Stanley, R.: Tools for requirements management: a comparison of telelogic doors and the hive. Tech. rep., Defence Science and Technology Organisation Edinburg (Australia) Information Networks DIV (2006)
7. Casamayor, A., Godoy, D., Campo, M.: Mining textual requirements to assist architectural software design: a state of the art review. Artificial Intelligence Review **38**(3), 173–191 (2012)
8. Conversy, S.: Unifying textual and visual: A theoretical account of the visual perception of programming languages. In: Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, pp. 201–212. ACM (2014)
9. Craik, F.I.: Aging and memory: Attentional resources and cognitive control (2019)
10. Cruz, S., da Silva, F.Q., Capretz, L.F.: Forty years of research on personality in software engineering: A mapping study. Computers in Human Behavior **46**, 94–113 (2015)

11. Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems. Commun. ACM **31**(11), 1268–1287 (1988)
12. De Vries, R.E., Van den Hooff, B., De Ridder, J.A.: Explaining knowledge sharing: The role of team communication styles, job satisfaction, and performance beliefs. Communication research **33**(2), 115–135 (2006)
13. Dobing, B., Parsons, J.: How UML is used. Communications of the ACM **49**(5), 109–113 (2006)
14. Donnellan, M.B., Oswald, F.L., Baird, B.M., Lucas, R.E.: The mini-ipip scales: tiny-yet-effective measures of the big five factors of personality. Psychological assessment **18**(2), 192 (2006)
15. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, pp. 285–311. Springer (2008)
16. Ellis, P.D.: The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results. Cambridge University Press (2010)
17. Falessi, D., Briand, L.C., Cantone, G., Capilla, R., Kruchten, P.: The value of design rationale information. ACM Transactions on Software Engineering and Methodology (TOSEM) **22**(3), 21 (2013)
18. Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering **23**(1), 452–489 (2018)
19. Gralha, C., Damian, D., Wasserman, A.I.T., Goulão, M., Araújo, J.: The evolution of requirements practices in software startups. In: Proceedings of the 40th International Conference on Software Engineering, pp. 823–833. ACM (2018)
20. Graziotin, D., Wang, X., Abrahamsson, P.: Do feelings matter? on the correlation of affects and the self-assessed productivity in software engineering. Journal of Software: Evolution and Process **27**(7), 467–487 (2015)
21. Guastello, S.J.: Creative problem solving groups at the edge of chaos. The Journal of Creative Behavior **32**(1), 38–57 (1998)
22. Hedges, L.V.: Distribution theory for glass's estimator of effect size and related estimators. journal of Educational Statistics **6**(2), 107–128 (1981)
23. Heijstek, W., Kuhne, T., R.V. Chaudron, M.: Experimental analysis of textual and graphical representations for software architecture design. In: International Symposium on Empirical Software Engineering and Measurement, pp. 167–176. IEEE (2011)
24. Hölldobler, K., Rumpe, B., Wortmann, A.: Software Language Engineering in the Large: Towards Composing and Deriving Languages. Computer Languages, Systems & Structures **54**, 386–405 (2018)
25. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering **5**(3), 201–214 (2000)
26. Hutchinson, J., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Science of Computer Programming **89**, 144–161 (2014)
27. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: Proceedings of the 33rd international conference on software engineering, pp. 471–480. ACM (2011)
28. Jarboe, S.: Procedures for enhancing group decision making. Communication and group decision making pp. 345–383 (1996)
29. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting experiments in software engineering. In: Guide to advanced empirical software engineering, pp. 201–228. Springer (2008)
30. Jolak, R., Ho-Quang, T., R.V. Chaudron, M., R.H. Schiffelers, R.: Model-based software engineering: A multiple-case study on challenges and development efforts. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 213–223. ACM (2018)
31. Jolak, R., Wortmann, A., R.V. Chaudron, M., Rumpe, B.: Does distance still matter? insights from revisiting collaborative distributed software design. IEEE Software (2018)
32. Karlström, D., Runeson, P.: Integrating agile software development into stage-gate managed product development. Empirical Software Engineering **11**(2), 203–225 (2006)

33. Kauffeld, S., Lehmann-Willenbrock, N.: Meetings matter: Effects of team meetings on team and organizational success. Small Group Research **43**(2), 130–158 (2012)
34. Koo, T.K., Li, M.Y.: A guideline of selecting and reporting intraclass correlation coefficients for reliability research. Journal of chiropractic medicine **15**(2), 155–163 (2016)
35. Kortum, F., Klünder, J., Schneider, K.: Don't underestimate the human factors! exploring team communication effects. In: International Conference on Product-Focused Software Process Improvement, pp. 457–469. Springer (2017)
36. Liebel, G., Marko, N., Tichy, M., Leitner, A., Hansson, J.: Assessing the state-of-practice of model-based engineering in the embedded systems domain. In: International Conference on Model Driven Engineering Languages and Systems, pp. 166–182. Springer (2014)
37. Liskin, O.: How artifacts support and impede requirements communication. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 132–147. Springer (2015)
38. McManus, M.M., Aiken, R.M.: Monitoring computer-based collaborative problem solving. Journal of Interactive Learning Research **6**(4), 307 (1995)
39. Meade, M.E., Wammes, J.D., Fernandes, M.A.: Drawing as an encoding tool: Memorial benefits in younger and older adults. Experimental aging research **44**(5), 369–396 (2018)
40. Meliá, S., Cachero, C., Hermida, J.M., Aparicio, E.: Comparison of a textual versus a graphical notation for the maintainability of mde domain models: an empirical pilot study. Software Quality Journal **24**(3), 709–735 (2016)
41. Moody, D.L.: The" physics" of notations: a scientific approach to designing visual notations in software engineering. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering, vol. 2, pp. 485–486. IEEE (2010)
42. Palomba, F., Panichella, A., De Lucia, A., Oliveto, R., Zaidman, A.: A textual-based technique for smell detection. In: 2016 IEEE 24th International Conference on Program Comprehension (ICPC), pp. 1–10 (2016)
43. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of agile practices on communication in software development. Empirical Software Engineering **13**(3), 303–337 (2008)
44. Ramos, A.L., Ferreira, J.V., Barceló, J.: Model-based systems engineering: An emerging approach for modern systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**(1), 101–111 (2012)
45. Ray, J.J.: The Construct Validity of Balanced Likert Scales. The Journal of Social Psychology **118**(1), 141–142 (1982)
46. Robillard, M.P., Marcus, A., Treude, C., Bavota, G., Chaparro, O., Ernst, N., Gerosa, M.A., Godfrey, M., Lanza, M., Linares-Vásquez, M., et al.: On-demand developer documentation. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 479–483. IEEE (2017)
47. Rumpe, B.: Agile Modeling with UML: Code Generation, Testing, Refactoring. Springer International (2017)
48. Rus, I., Lindvall, M., Sinha, S.: Knowledge management in software engineering. IEEE software **19**(3), 26–38 (2002)
49. Santos, A., Gómez, O.S., Juristo, N.: Analyzing families of experiments in SE: a systematic mapping study. IEEE Transactions on Software Engineering (2018)
50. Sharafi, Z., Marchetto, A., Susi, A., Antoniol, G., Gueheneuc, Y.G.: An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In: Program Comprehension (ICPC), 2013 IEEE 21st International Conference on, pp. 33–42. IEEE (2013)
51. Shatnawi, A., Seriai, A., Sahraoui, H., Al-Shara, Z.: Mining software components from object-oriented apis. In: International Conference on Software Reuse, pp. 330–347. Springer (2015)
52. Singer, J., Vinson, N.G.: Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering **28**(12), 1171–1180 (2002)
53. Soller, A.: Supporting social interaction in an intelligent collaborative learning system. International Journal of Artificial Intelligence in Education (IJAIED) **12**, 40–62 (2001)
54. Storey, M.A., Zagalsky, A., Figueira Filho, F., Singer, L., German, D.M.: How social and communication channels shape and challenge a participatory culture in software development. IEEE Transactions on Software Engineering **43**(2), 185–204 (2017)

55. Tang, A., Aleti, A., Burge, J., van Vliet, H.: What makes software design effective? Design Studies **31**(6), 614–640 (2010)
56. Tang, A., Babar, M.A., Gorton, I., Han, J.: A survey of architecture design rationale. Journal of systems and software **79**(12), 1792–1804 (2006)
57. Tang, A., Bex, F., Schriek, C., van der Werf, J.M.E.: Improving software design reasoning–a reminder card approach. Journal of Systems and Software **144**, 22–40 (2018)
58. Tang, A., Tran, M.H., Han, J., Van Vliet, H.: Design reasoning improves software design quality. In: International Conference on the Quality of Software Architectures, pp. 28–42. Springer (2008)
59. Tversky, B.: Multiple models. in the mind and in the world. Historical Social Research/Historische Sozialforschung. Supplement (31), 59–65 (2018)
60. Völter, M.: Md*/dsl best practices update march 2011. Update (2011)
61. Wagner, S., Fernández, D.M.: Analyzing text in software projects. In: The Art and Science of Analyzing Software Data, pp. 39–72. Elsevier (2015)
62. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
63. Wolf, M., Petridis, M., Ma, J.: Using structural similarity for effective retrieval of knowledge from class diagrams. In: International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 185–198. Springer (2013)
64. Wortmann, A., Barais, O., Combemale, B., Wimmer, M.: Modeling languages in industry 4.0: an extended systematic mapping study. Software and Systems Modeling (2019)
65. Wu, J., Graham, T.C.N., Smith, P.W.: A study of collaboration in software design. In: 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings., pp. 304–313 (2003)