2021-09

How Templated Requirements Specifications Inhibit Creativity in Software Engineering

R. Mohanani, P. Ralph, B. Turhan, Mandić Vladimir

IEEE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

How Templated Requirements Specifications Inhibit Creativity in Software Engineering

Rahul Mohanani, Paul Ralph, Burak Turhan, *Senior Member, IEEE* and Vladimir Mandić, *Member IEEE*

Abstract—Desiderata is a general term for stakeholder needs, desires or preferences. Recent experiments demonstrate that presenting desiderata as templated requirements specifications leads to less creative solutions. However, these experiments do not establish how the presentation of desiderata affects design creativity. This study, therefore, aims to explore the cognitive mechanisms by which presenting desiderata as templated requirements specifications reduces creativity during software design. Forty-two software designers, organized into 21 pairs, participated in a dialog-based protocol study. Their interactions were transcribed and the transcripts were analyzed in two ways: (1) using inductive process coding and (2) using an a-priori coding scheme focusing on fixation and critical thinking. Process coding shows that participants exhibited seven categories of behavior: making design moves, uncritically accepting, rejecting, grouping, questioning, assuming and considering quality criteria. Closed coding shows that participants tend to accept given requirements and priority levels while rejecting newer, more innovative design ideas. Overall, the results suggest that designers fixate on desiderata presented as templated requirements specifications, hindering critical thinking. More precisely, requirements fixation mediates the negative relationship between specification formality and creativity.

Index Terms—Cognitive bias, software design, fixation, critical thinking, requirements, requirements engineering, protocol analysis.

1 Introduction

Different organizations initiate new software projects in many different ways. For teams developing consumer applications and enterprise systems, however, the initiation process often seems to include:

- speaking with prospective users and other stakeholders about their needs, wants, preferences, etc.;
- synthesizing stakeholders' opinions into some documents;
- determining the main features that the product will have; and
- creating mock-ups and diagrams illustrating the main features and user interfaces.

These activities can be sequential, parallel, in different orders and performed by the same or different people. Nevertheless, it raises an obvious question—what kind of documents are best for synthesizing stakeholders' opinions? Different areas of research seem to have reached different conclusions.

The more positivist side of requirements engineering (RE) research tends to assume that software projects have discoverable and documentable requirements, and that understanding these requirements is critical for designing good software systems [1], [2]. It seeks to elicit unambiguous, consistent, complete, feasible, traceable and verifiable requirements [3]. Good requirements specifications should lead to good software designs [4] because meeting requirements is what 'good' means.

Contrastingly, the more naturalistic side of RE, as well as research in human–computer interaction, user–centred design, and the interdisciplinary design literature tends to assume that:

- software projects do *not* have discoverable and documentable requirements (cf. [5]);
- stakeholders do not even have stable, retrievable preferences (cf. [6]); and
- products have numerous stakeholders who do not agree on the problem(s) to solve or how the product should solve them (cf. [7]).

Forcing vague, unstable, conflicting preferences into unambiguous, consistent requirements specifications encourages designers to converge prematurely on oversimplified problems and inappropriate solutions [8]. Eliciting templated requirements specifications should therefore lead to designs that satisfy contracts but not users, which is antithetical to user–centred design.

Our previous work showed that presenting a set of desiderata as templated requirements specifications led to less creative product designs than presenting exactly the same desiderata as uncertain ideas. [9], [10]. However, experiments like these are not suitable to explore cognitive mechanisms underlying causal effects, so we have evidence that the presentation of desiderata affects creativity but we don't know how. This raises the following research question.

> ***Research question:*** *How do fixation and critical thinking explain reduction in design creativity when desiderata are presented as templated requirements specifications?*

Here, *fixation* means the tendency of the designers to pay excessive and undue attention to the given problem by

- *R. Mohanani is with fortiss, Munich, Germany. E-mail: rahul.mohanani@gmail.com, mohanani@fortiss.org*
- *P. Ralph is with the Faculty of Comp. Sci., Dalhousie Univ., Canada. E-mail: paul@paulralph.name*
- *B. Turhan is with M3S Group, Univ. of Oulu, Finland. E-mail: turhanb@computer.org*
- *V. Mandić is with the Faculty of Tech. Sci., Univ. of Novi Sad, Serbia. E-mail: vladman@uns.ac.rs*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

readily converging on an available or known solution (cf. [11]); *critical thinking* is defined as "disciplined thinking that is clear, rational, open-minded, and informed by evidence" [12]. *Design creativity*, for our purposes, denotes the originality and practicality of new product concepts. *Desiderata* are properties of a real or imagined system that are wanted, needed or preferred by one or more project stakeholders [13]. We use this term because it helps us remember that the set of things a stakeholder wants and the set of things needed for a system to succeed do not always coincide. While *requirements specification* is defined as "a statement that identifies a capability or function that is needed by a system in order to satisfy its customer's needs" [14], *templated requirements specification* (TRS) is requirements specification written in a specific syntactic structure using a restricted (controlled) natural language [15], in this case, for example, "*The system shall facilitate diet planning*".

Next, we review existing literature (Section 2). Then, we describe our research design including data collection and analysis (Section 3), followed by the results (Section 4). Section 5 discusses the theoretical framework and summarizes the study's implications and limitations. Section 6 concludes the paper with a summary of its contributions.

2 Background

This section summarizes the major concepts involved in this study: desiderata, task structuring, creativity and fixation.

2.1 The concept of desiderata

In the most extreme positivist view of RE, requirements are a property of the environment, which are motivated by the desires, wants and needs of the stakeholders [16], [17]. Requirements analysts elicit them, and success means fulfilling them. Since social reality is real and objective, requirements exist in the world, waiting to be discovered. Project success means meeting and satisfying the requirements.

In the naturalistic view of RE and the constructivist view of product design, requirements do not exist in an objective reality [5], [18], [19] waiting to be discovered. Reality is socially constructed. Stakeholders usually have unstable, unreliable, conflicting desiderata [6], [20] and use different processes and representations to express their opinions. Research in RE has tried to organize and manage these conflicts and inconsistencies among stakeholders to elicit requirements. Some of the techniques include the ViewPoints—a framework that facilitates capturing, representing and organizing multiple stakeholders' viewpoints and perspectives [21], conflicts and goal–modelling [22] and AbstFinder—a tool that helps finding abstractions and textual ambiguity in natural language text [23]. Project success then means delivering benefits to stakeholders [24]. Fundamentally, RE is about establishing a balance between the positivist approach where desiderata are considered as a singular truth embodied in the form of a formal specification, and the naturalistic approach where requirements are a product of the conflicts and contradictory viewpoints of the stakeholders involved (cf. [25]).

Practically, the main difference in the above discussion is what happens when a stakeholder demands that the system has some property or fills some need. In the positivist view, the optative speech act of demanding manifests a requirement [26]. In the multi-perspective constructivist view, the stakeholder's demand is just an opinion of note; the demanded property may be a necessary condition for success, irrelevant, or even prevent success. This manuscript assumes the latter.

2.2 Task structuring

Problems are often conceptualized on a spectrum of well-structured to ill-structured. "Well-structured problems are constrained problems with correct or convergent solutions that require the application of a limited number of rules and principles within well-defined parameters; whereas, ill-structured problems possess multiple solutions and fewer parameters that are less manipulable and contain uncertainty about the concepts, rules, and principles that are necessary for the solution, the way they are organized and which solution is best" [27, p. 65]. A body of empirical research shows that task structure is negatively associated with design performance (cf. [28] for summary). "Over-concentration (over-structuring) on problem definition does not necessarily lead to successful design outcomes" [29, p. 439] for at least four reasons:

1) less specific goals reduce cognitive load [30], which leads to more learning;
2) less specific task framing results in more creative solutions [31];
3) designers often fixate on experience [32] or on an initial set of ideas [33]; and
4) designers often process whatever little information they have and quickly assimilate it into the problem schema, improving their understanding of the problem [34].

Perhaps unsurprisingly, then, our recent experiments showed that presenting desiderata as TRS reduced creativity [9], [10]. We hypothesized that presenting desiderata as TRS triggers a specific cognitive bias, which we call requirements fixation. Fixation broadly refers to the tendency to "disproportionately focus on one aspect of an event, object, or situation, especially due to self-imposed or imaginary obstacles" [11, p. 5]. Requirements fixation, then, is the tendency to attribute undue confidence and importance to desiderata presented as TRS. We use the term requirements fixation to allude to similarity to *design fixation*: the well-established tendency for designers to generate solutions very similar to given examples [35] or existing artifacts [36].

Although the precise mechanism by which increasing task structure reduces design performance and creativity remains unclear, design expertise seem to moderate the relationship. Expert designers tend to resist initial problem framing (e.g. given TRS) and proceed via an improvised, solution-focused approach [37]. Expert designers consider all problems ambiguous and ill-structured, focusing on solution generation rather than analysing the given problem [29], [38]. On the other hand, novice designers often treat ill-structured problems as well-structured, thereby compromising the potential for creative solutions [39]. However, recent research suggests that novice designers as compared to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

more experienced designers, are less fixated on the problem domain and, hence, are able to generate highly creative solutions [40], [41].

2.3 Creativity

RE is a creative process [42], [43], where analysts and multiple stakeholders collaborate to make sense of a problematic situation and conceptualize a common mental model of a possible system [44]. In RE, creativity enhancing workshops (e.g., [45]) are extensively used to provide clarity for requirements identification [46], [47] and generate novel and creative requirements [48], [49]. In these workshops, creativity is often linked with divergent thinking [50], i.e., exploring multiple and diverse solutions to a given problem. While Brainstorming helps in generating most number of requirements, Hall of Fame approach (viz., [51]) helps in developing multiple creative requirements [52]. In another study, interactive collaboration techniques employed during the workshops helped in generating more creative requirements [48]. In a nutshell, requirements can be understood as entities that encapsulate the results of creative thinking about the system being developed [53].

Research in RE also focuses on ways to discover and generate creative requirements by leveraging the way in which problem situations and early ideas are represented. One study preferred using user stories to explore novel ideas, which were then used to measure the personality traits and creative potential of the participants influencing their creative abilities [54]. In another study, high-level goals, represented as goal models (viz., [55]), were combined with creativity enhancing techniques to explore and generate creative requirements [56]. Integrating the concept of combinational creativity (viz. [57]) with use cases (e.g., [58]) and creativity enhancing framework (e.g., [59]) are also used to discover and develop creative requirements.

Creativity, itself, is a poorly-understood, multi-dimensional construct [60], [61]. However, creativity *research* can be organized into the "Six P's" (viz., [62], [63]), as follows: 1) creativity's underlying cognitive *process*; 2) the creative *product*; 3) the *person* (or personality) doing the creative work; 4) the *place* (or context) of the creative work; 5) stimulating creative thinking (or persuasion) and 6) improving creative *potential*.

Here, we are primarily concerned with *product* creativity. Product creativity is entails two dimensions: (1) novelty or originality [64], [65] and (2) practicality or usefulness [66], [67]. Therefore, we conceptualize creativity as the ability of a designer to choose both novel and practically useful features, graphical elements and aesthetic properties of a software system.

2.4 Fixation

Cognitive biases are systematic deviations from optimal reasoning [68]. Since software engineering involves lots of reasoning, cognitive biases help to explain common problems in software design [69], [70], testing [71], [72], requirements engineering [73], [74] and project management [75].

Fixation is a cognitive bias in which "blind adherence to a set of ideas or concepts limit[s] the output of conceptual designs" [32]. Several experiments have demonstrated design fixation—the tendency for designers to generate solutions very similar to given examples [32], [35] or existing artifacts [36]. The cognitive mechanisms underlying design fixation are not well understood. However, [35] suggest that designers may fixate on a known but limited set of ideas or an existing body of knowledge, and classify design fixation into three broad categories:

1) Unconscious adherence—designers depend too heavily on ideas encoded in long term memory, sometimes due to heavy load on working-memory.
2) Conscious blocking—designers dismiss new ideas due to over-dependence and confidence on their old ideas or past experience.
3) Intentional resistance—designers intentionally resist new ideas to designs that were previously successful.

Meanwhile, design fixation is moderated by several factors:

1) The domain; for instance, mechanical engineers fixate more than industrial designers [76];
2) examples—common examples causes more fixation than unusual or rare examples [77];
3) defixating instructions—explicit instructions to avoid features of given or existing artifacts [78];
4) providing good quality examples than flawed or no examples lead to better design performance [79];
5) product dissection activities [80]; and
6) physical prototyping [81].

SE research shows that inconsistency in requirements specifications may reduce premature commitment [82]. More generally, the way a task is communicated or presented may also cause fixation [83], [84]. This is related to the framing effect—"the tendency to give different responses to problems that have surface dissimilarities but are formally identical" [85, p. 88]. Desiderata can be presented in many different forms, including TRS, personas, scenarios, use cases and requirements statements. We can think about these forms as different ways of framing a design task, and task framing affects design performance [9], [10], [86].

While, the underlying cognitive mechanisms that reduces creativity of design concepts due to TRS are not yet well explored, our recent experiments in SE (e.g., [9], [10]) suggest a typical behavior where designers tend to shut down their creative potential by further inflating the high importance and confidence connoted by the TRS.

3 Research Design

3.1 Dialog-based protocol analysis

To answer our research question, we need real-time insight into software designers' cognitive processes. Think-aloud protocol analysis is a research methodology in which participants verbalize thought sequences. Researchers analyze participants' words for insight into their thinking. Researchers assume that "any concurrent verbalization produced by a subject while solving a problem is a direct representation of the cognitive functioning (i.e., mental processes) of the subject's working memory" [87].

However, verbalizing our thought process probably changes those thought processes in imperceptible ways.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

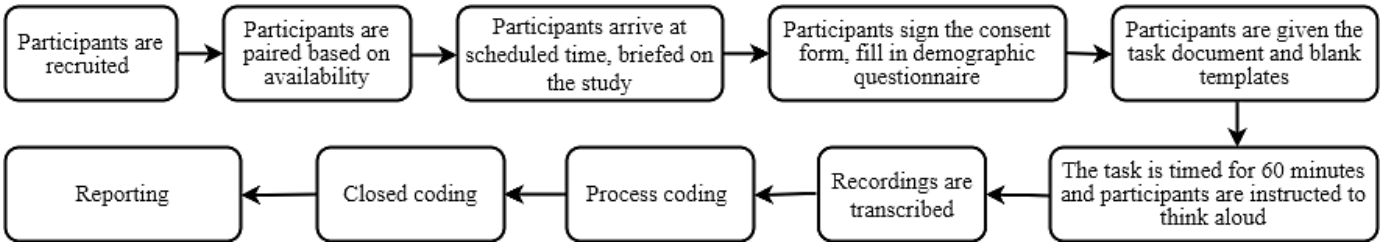

Fig. 1. Overview of the data collection and analysis process

We can mitigate this limitation using dialog-based protocol analysis [88], in which participants work in pairs or groups, and we analyze their natural dialog instead of a forced monologue.

Protocol analysis is common in design studies [89], psychology (e.g. [87]), medicine (e.g. [90]) and software engineering [88], [91]. We see protocol studies as most consistent with a critical realist philosophy of science (cf. [92]).

3.2 Purpose and scenario

The purpose of this study is to determine why presenting desiderata as TRS reduces design creativity. To do so, we observe participants in a simulation of a situation where a software team is given a set of TRS and then asked to design an appropriate application. Although RE and high-level designing are increasingly merged (e.g., [93]–[95]), such situations arise often in outsourcing arrangements (where the outsourcer provides a questionable requirements specification and expects the team to develop mock-ups without much access to prospective users or client representatives). In our experience, many software teams that have stakeholder access still have specifications forced upon them by clients, management, marketing or other stakeholders. Fig. 1 summarizes our research protocol.

An alternative question—whether modeling desiderata as TRS harms creativity in teams that both create the TRS and then design the product—is potentially fruitful avenue of future work. We did not attempt such a simulation because it is inconsistent with the prior experiments we are attempting to re-examine and entails numerous unresolved methodological problems [96, section 1.3].

3.3 Participants and pairing

We recruited a convenience sample of 18 professional software developers (14 men, 4 women) from Company X, which develops web and mobile applications for government agencies, corporations and educational institutions. Company X has 30 employees, with a typical project duration of 2-3 years. We selected Company X because it was willing to participate due to close ties to one of the authors. These participants had a mean age of 31 years ($\sigma = 5.65$).

Meanwhile, we recruited 24 post-graduate students (21 male, 3 female) enrolled in the information processing science program at the first author's university. Student participants had a mean age of 23 years ($\sigma = 6.07$). They received extra credit in one of their courses for participating.

While professional participants had a mean work experience of 5.6 years, student participants had a mean work experience of 2.3 years. All participants had at least 1 year of experience in software design. However, none of the participants had any experience with developing health and fitness applications—the domain used here. Participants were paired based on availability. Each pair comprised either two professionals or two students.

3.4 Execution of the study

The study was approved by the University of Auckland Human Participants Ethics Committee (UAHPEC), New Zealand. We facilitated and supervised the data collection from September to December, 2016, at Company X's office for the professionals and at the students' university. Every participant–pair was scheduled an individual session in a quiet room. On arrival, the study was described and participants signed a consent form and completed a demographic questionnaire.

We used the same task as our previous experiments [9], [10]. The task document listed 25 desiderata presented as TRS and organized into five priority levels: high, high–medium, medium, medium–low and low. Each TRS began with "The system shall" (consistent with [3]) and phrased as, for example, "*The system shall measure calorie intake*", "*The system shall recommend activities*" and so on. Crucially, many of the desiderata presented in this task are ill-considered, inconsistent or over-complicated. The TRS was compiled to engender skepticism.

The participants were also given identical design templates comprising blank, mobile screen–sized boxes in portrait and landscape orientations with space for written explanations. Participants were then asked to generate conceptual designs of a health and fitness mobile application. Participants could use as many templates as needed. The participants were encouraged to discuss their thoughts while creating designs.

The first author acted as facilitator for both groups. Sessions were limited to 60 minutes at Company X's request. Students were also limited to 60 minutes for consistency. During each session, the facilitator took notes, reminded participants to stick to English and prompted participants who made design moves without discussion.

We piloted the study once to check the recording equipment. No subsequent changes were made to the task or the study procedure. All task documents including the TRS and their qualities are available in our replication package (see Section 7).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

TABLE 1
Summary of process coding analysis

Theme	Example labels	Example quotations and dialogues
Making design moves	Discussing design moves	"OK, you're on the intake page or seeing just the intake, and then you will have a small button." "A-ha, so it could be like.." "icon, something like this.." "Yeah" (P15)
	Generating multiple design options	"We can add into the 'diet screen' an option to recommend." "Or rather it should be over here...you select and you search." (P2)
	Making moves	"We can have help me button to propose exercise. And eat this meal or skip, best practice to reach goal" "The BMR is calculated after this segment screen you have all the information" (P2)
Uncritically accepting	Accepting features of existing examples	"You have all the history, in one click. And here you have, most of the fitness apps like FitBit have this, so, why not use it?" "Yeah, okay" (P2)
	Accepting priorities	"If we have to design an app that shows all these five requirements that are on a high priority, we would probably want to have all the data on one screen" (P16)
	Accepting requirements	"So now we have requirement number 5, to provide workout history and performance analysis" "Yeah, that could be on the main page that given history and performance analysis" (P11)
Rejecting	Rejecting design moves	"I just thought we could split the screen and it shows you the current activity and any previous activities" "No, it can still show the previous one. Let's stick to this way for now" (P16)
	Rejecting requirements due to no knowledge	"So on to requirement 10, we shouldn't consider much on this one?" "Yeah.. it's more like a system requirement, I don't know...let's move on" (P13)
	Rejecting specifications due to time constraints	"Well the analysis screen would be the most complicated one. I don't think we have time for that now" "Yeah, let's check other ones" (P6)
Grouping	Grouping seemingly similar requirements	"So recommend recipes and recommend workouts and recommend diet food, all could be groped as one. They all say recommend" "Yeah, okay" (P11)
	Grouping as input-type data	"We put a dashboard feature" "like for input data" "Yeah, let's group them as input data" (P1)
	Grouping requirements of same priority level	"The first two high priority tasks of measuring calorie intake and what user eats and drinks is the same thing" "That's true" "So those are kind of easy to put together..." (P15)
Questioning	Questioning priority levels	"The system must allow the user to plan workout...this isn't very important. Shall track speed and distance..this is very important" "Me too" (P21)
	Questioning existing examples	"They FitBit app have sensors and they can do precise measuring.. running is based on GPS." "Yeah, maybe we should do something else. GPS is not always good and reliable." (P3)
	Questioning requirements	"We don't have to include all the. But, why would someone, why would you listen to music?" (P2)
Assuming	Assuming on behalf of the users	"How we will measure calories in it. Will user write?" "Maybe he will write" "Okay, I suppose" (P9)
	Assuming relative importance	"Now, what is the most important thing to the user, to know the amount of calories or what he actually ate? I think the calories are the most important than other ones here" (P8)
	Assuming time limitations	"This cannot be done in one hour." "No definitely not. I'm not sure, if I have an idea" (P6)
Considering quality criteria	Consistency	"Have a graph that tells the amount of workout and sleep" "That maintains consistency" (P9)
	Usability	"I think this screen must be very easy and quick to use. It just can't be a massive calendar" (P12)
	Responsiveness	"Or you can make it like, responsiveness, more like on smaller screens to change the design, so I make it for that one." "Maybe we could make it smaller" (P5)

3.5 Data collection and analysis

The sessions were transcribed by the first author. Although we did not correct participants' grammar or malapropisms, we removed verbal static (e.g., "um", "ah", "uh"). We refer to each transcript by a unique identifier starting with a 'P'. P1–P9 are the professionals; P10–P21 are the students.

We envisaged the data analysis in two phases: 1) inductive process coding to explore the cognitive mechanisms used by the participants; 2) deductive closed coding using concepts from existing literature to re-analyze the data through a specific theoretical lens. We used NVIVO (www.qsrinternational.com) to organize, analyze and visualize the qualitative data. The coding process is briefly described as follows:

1) The first author performed process coding of all the transcripts (see Section 3.5.1).
2) The second and the third author audited process coding.
3) The auditing resulted in renaming some of the codes (e.g. *doing design* was renamed to *making moves*); some codes were combined together or rejected altogether (e.g. *avoiding options* and *rejecting moves* were combined to *rejecting design moves*).
4) The first and the second author performed closed coding of one transcript together to ascertain the coding scheme (see Section 3.5.2).
5) The first author then coded the rest of the transcripts.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

6) The second and the third author audited closed coding.
7) The auditing resulted in minor changes such as adding new codes to the instances of fixation or critical thinking (e.g. *rejecting design moves* was added to fixation).

3.5.1 Process coding

We began by analyzing the data using inductive process coding [97]. That is, we coded each transcript line-by-line using gerunds (i.e., words ending with '-ing'). Each assigned label reflected the action contained in dialogues that shared similar characteristics (e.g., accepting requirements, discussing design moves). As the analysis progressed, some labels were reworded, subsumed by other similar labels or dropped. All codes that conveyed a particular process (i.e., action) were further categorized together to form themes, where a theme was seen as a high level conceptualization of multiple labels grouped together [98]. The saturation point was reached by the 14th transcript, i.e. no new labels or themes emerged on from the remaining seven transcripts.

3.5.2 Closed coding

Our previous experiments suggested the tendency of participants to get fixated on desiderata when presented as TRS. The effects of fixation can be minimized by a critical evaluation of the problem situation. Any attempt to critically evaluate the TRS should help participants to avoid fixating on the given TRS. Therefore, we applied an a priori coding scheme to compare instances of fixation against instances of critical thinking. Here, fixation refers to instances where participants: 1) accept aspects of the task (i.e., requirements, priority levels) without any discussion or reflection; 2) adopt properties of known examples without any discussion or reflection; or 3) reject, without any discussion or reflection, new ideas that diverge from given task structure or known examples. Critical thinking meanwhile refers to instances where participants critically evaluate or deviate from task parameters or known examples. In other words, if participants question something, but then accept it, we label it as critical thinking. We then counted these instances and compared.

4 Results

This section presents the results from both analyses.

4.1 Process coding

Process coding produced seven themes, each of which we interpret as a distinct cognitive activity. Table 1 summarize the evidence for each themes, while the complete analysis is available in our replication package (see section 7).

4.1.1 Making design moves

A design move is a change to a design description [99]. *Considering and making design moves* was the participants' most frequent activity. We found a total of 48 instances in fourteen groups where participants willingly tried to come up with multiple design ideas, reflected on those ideas and then made a move by selecting the most optimum one. However, participants in nine groups made design moves only intending to satisfy all the given requirements without assessing or reflecting on them. Out of these, six groups only tried to meet the requirements prioritized as high or high–medium.

4.1.2 Uncritically accepting

We observed participants *uncritically accepting* their initial design ideas and aspects of the task (e.g., requirements, priority levels) without any discussion or reflection. Participants were keen to adopt and force features of existing examples into their design concepts without assessing the existing designs. We observed imbalances in the pairs where Partner A would immediately accept Partner B's ideas, while Partner B would unthinkingly reject Partner A's ideas whenever they diverged from Partner B's ideas.

4.1.3 Rejecting

Another pattern that emerged was the tendency of participants to explicitly *reject* any requirements or design ideas for various reasons (e.g., ambiguity, difficulty). Moreover, participants appeared reluctant to satisfy requirements prioritized as low or medium–low and requirements about which they had no knowledge. Participants would either temporarily ignore a high-priority requirement, or would permanently dismiss a requirement. When participants reject requirements, they often did so without any discussion, reflection or evaluation. Thirteen pairs explicitly rejected any idea or design move that diverged from their first design concept. We observed both uncritically accepting initial ideas (as discussed above) and uncritically rejecting new ideas that diverge from initial ideas.

4.1.4 Grouping

Fourteen of the pairs made sense of the desiderata by *grouping* requirements they perceived as similar. Out of these, eight groups were based on the priority levels provided, others on other sorts of similarity. For example, grouping given requirements as non-functional features or as pop–up notifications. Subsequent design moves appear to be informed by these groups. The tendency of participants to focus on grouping only high priority requirements while avoiding the low priority ones (we observed a total of 47 such instances) can be related to participants' uncritical acceptance of priority levels and task presentation more generally.

4.1.5 Questioning

While participants often uncritically accepted aspects of the task (see section 4.1.2), they questioned others. Here, *questioning* refers to critically appraising something. Questioning is related to but distinct from rejecting. Sometimes participants questioned something before rejecting it; other times participants questioned something before accepting it; and other times participants rejected something without really questioning it first. Typically, one participant would raise doubts about a something (e.g., requirements, priority levels, existing examples). The pair would then discuss and come to a consensus about accepting or rejecting the concept. Participants also backtracked on their earlier design moves based on their evolving understanding of the task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

4.1.6 Assuming

Eleven pairs made explicit *assumptions* about the task. These assumptions were basically cognitive shortcuts that participants used to help them create designs easily with minimal information processing. We consider these assumptions as a deviation from the real or the observable facts. Participants appeared to speculate and derive at rather specific conclusions about how they perceived the requirements, instead of making actual sense of the problem situation. Five groups unreasonably perceived high priority requirements as more important than other low priority ones (e.g., counting calories as more important than recommending workouts); while other groups would over-estimate the effort required by creating self-imposed time-constraints. Participants also assumed a requirement (e.g., workout recommendation) was non-functional and jumped to conclusions about the complexity of initial design ideas.

4.1.7 Considering quality criteria

While planning the designs, participants would often *express the need* for certain aesthetic qualities for their solution designs. Participants in nine groups explicitly tried to change or alter their design moves for various quality criteria including usability, consistency of user experience, system responsiveness, speed, stability and aesthetics.

4.2 Closed Coding

This section presents the results of our closed coding. We identified 1006 instances of *fixation* compared to 298 instances of *critical thinking*. Table 2 presents the list of labels classified in each category and the corresponding number of instances of each label. Below, we briefly discuss each category and interpret our findings.

4.2.1 Fixation

All of the participants showed a tendency to agree and to accept instantaneously aspects of the task. We found 178 instances of participants accepting requirements without question and 145 cases of accepting priority levels without question. Participants appeared to accept task structure without any discussion of or reflection on the importance or the validity of the given desiderata. We also found 41 instances where pairs explicitly rejected new design ideas because they diverged from the initial design ideas.

Moreover, we found 517 instances where participants expressed complete confidence and extensively favored their initial (i.e., early) ideas by avoiding any speculation, discussion or reflection. We observed 79 cases across 19 pairs where participants tried to conceptualize their solution designs based on either a successful example or their previous experience. For example, "*And also, kind of integration with Spotify or, another provider like Pandora like in other health fitness app I have come across*" "*We should do exactly that" (P1)*.

In 22 instances, participants said or implied that the system should satisfy *all* of the requirements; in ten instances participants said or implied that the system should satisfy at least all of the high and medium–high priority requirements. This is surprising because the requirements are intentionally dubious (as explained in Section 3.4).

4.2.2 Critical thinking

We found substantially fewer instances where participants attempted to think about the task parameters critically. In 199 instances pairs critically discussed design moves. However, these discussions were mainly about planning and organizing the way the application would look, and less about selecting the best option from multiple design options or assessing the importance of the given requirements. We observed eleven instances in six pairs of backtracking on their earlier design decisions. Such instances were characterized by participants reflecting on their initial design decision, followed by discussing alternative ideas and then selecting the one perceived as most appropriate. Some participants explicitly tried to generate multiple design options. We found very few instances where participants critiqued or reflected on the specific aspects of the task itself. Six pairs questioned priority levels with 14 instances; only two pairs expressed overall doubts about the requirements with four instances.

Similarly, we found nine instances of participants questioning the available technology and six instances of challenging existing examples. (Note: we did not provide examples; participants looked up examples on their own during the study.) Only three groups attempted to generate ideas that were not evident from the task materials, found in similar applications or generated by other groups. Only one pair expressed the need to consult the client to clarify specific ambiguous requirements.

4.2.3 Differences between students and professionals

Table 3 summarizes the differences between the professionals and the students. Professionals had more instances of both fixation and critical thinking than students; however, these differences are not statistically significant (Independent samples t-tests with effect size via Cohen's d; fixation: $t = 0.49, p = 0.63, d = 0.23 \pm 0.87$; critical thinking: $t = 0.91, p = 0.37, d = 0.42 \pm 0.89$). Moreover, the ratio of fixation to critical thinking [1] is almost identical (0.76 professionals vs. 0.78 for students). In other words, both students and professionals seem about equally susceptible to fixation.

Surprisingly, months of development experience is *positively* correlated with fixation (Pearson correlation; $r = 0.528; p = 0.014$)—see Fig. 2—but uncorrelated with critical thinking ($r = -0.93; p = 0.689$). In other words, more experienced developers were more prone to fixation. While these are post hoc tests on convenience samples, the results question the idea that fixation is limited to amateurs or that experience naturally mitigates it.

5 Discussion and Implications

5.1 Theory

Previous work [9], [10], [31] showed that presenting desiderata as TRS diminished creativity, which undermines SE success. The purpose of this study is to investigate *how* presenting desiderata as TRS diminishes creativity; that is, the

1. $\bar{F}/(\bar{F}+\bar{C})$ where $\bar{F}$ is the mean number of actions associated with fixation and $\bar{C}$ is the mean number of actions associated with critical thinking

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

TABLE 2
Closed coding analysis

Pairs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Total
Category 1: Fixation																						
Accepting early ideas	37	11	21	10	21	10	42	50	34	19	24	32	17	22	27	16	39	14	24	31	16	517
Accepting reqs	10	2	4	5	10	12	5	3	7	6	14	8	23	8	7	12	15	7	8	5	7	178
Accepting priorities	12	8	5	3	9	7	11	7	2	9	5	8	7	4	6	7	9	9	11	2	4	145
Accepting features of existing examples	3	6	12	-	3	2	2	5	3	3	4	7	3	5	4	2	4	-	2	3	6	79
Rejecting design moves	2	5	4	2	-	7	-	4	2	-	-	-	2	1	-	2	-	-	2	6	2	41
Trying to satisfy all reqs	1	3	2	1	-	3	-	4	2	-	-	-	2	-	-	-	-	-	4	-	-	22
One participant accepts others decision	-	-	-	-	-	5	-	-	-	-	-	-	3	-	-	-	-	-	6	-	-	14
Trying to satisfy high priority reqs	2	1	-	-	-	1	-	1	-	2	-	-	-	-	-	1	-	1	-	-	1	10
Total instances of fixation	67	36	48	21	43	47	60	74	50	39	47	55	57	40	44	40	67	31	57	47	36	**1006**
Category 2: Critical thinking																						
Discussing design moves	4	15	8	2	9	9	4	16	13	12	19	7	5	12	17	3	11	14	5	8	6	199
Generating multiple design options	-	9	5	1	2	1	6	5	3	-	-	5	-	-	-	2	2	1	-	4	2	48
Questioning priority levels	-	-	2	2	-	-	-	-	-	-	3	-	1	-	-	-	-	3	-	-	3	14
Backtracking on earlier decisions	-	-	-	-	2	-	1	3	-	2	-	-	-	1	-	-	-	-	-	-	2	11
Questioning available technology	-	2	-	-	1	-	1	-	-	-	-	-	-	1	1	-	-	2	-	1	-	9
Questioning existing examples	2	-	2	-	-	-	-	1	-	-	-	-	-	-	-	-	1	-	-	-	-	6
Generating innovative ideas	-	2	-	-	-	2	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	5
Questioning reqs	-	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4
Questioning client's needs	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
Total instances of critical thinking	6	30	21	5	14	12	12	25	17	14	22	12	6	14	18	5	14	20	5	13	13	**298**

TABLE 3
Differences between students and professionals

	Professionals	Students
Mean instances of fixation	49.67	46.67
Mean inst. of critical-thinking	15.87	13
Range (fixation)	21–74	31–67
Range (critical thinking)	5–30	5–22
Fixation ratio[1]	0.76	0.78

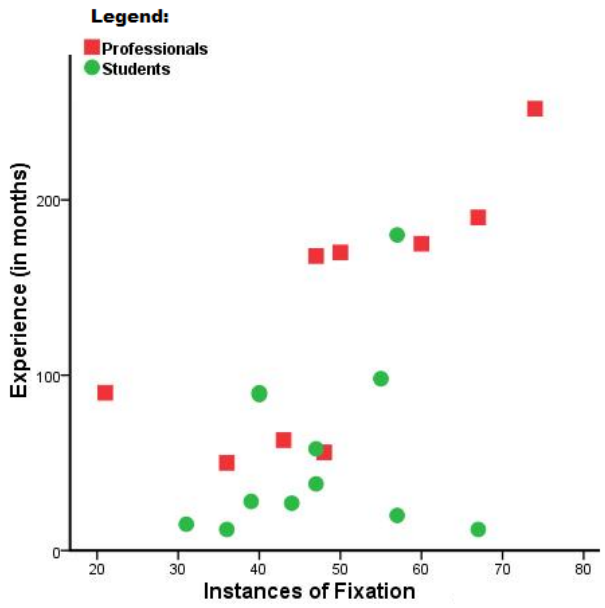

Fig. 2. Scatterplot of experience (months) vs. fixation (instances)

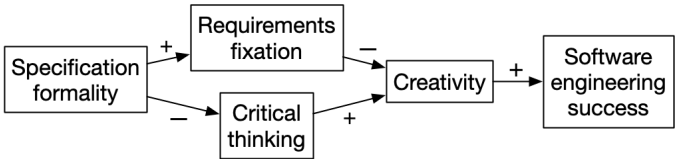

Fig. 3. Theoretical framework

Note: Boxes indicate constructs; arrows indicate causality; plus and minus signs indicate direction of effect.

cognitive mechanism mediating the previously established causal relationship. The results of this study suggest that providing designers with TRS induces requirements fixation and hinders critical thinking, thereby negatively affecting creativity (see [9], [10]); as shown in Fig. 3—see Table 4 for construct definitions.

The simulation reported above shows that participants given TRS have many more instances of fixation than critical thinking. Furthermore, we identified several indicative behaviours for both fixation and critical thinking—the labels in Table 2. For example, unthinking acceptance of TRS indicates requirements fixation; questioning the reasonableness of priority levels indicates critical thinking.

5.2 Implications

The interplay between fixation, critical thinking, creativity and the presentation of desiderata as TRS has numerous implications for SE professionals, researchers, and educators.

For professionals, the best way to present desiderata involves trade-offs among many criteria including clarity, understandability, flexibility, modifiability and, of course, creativity ([100], [101]). Where creativity is a priority,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

TABLE 4
Theoretical Concept Definitions

Concept	Definition
Specification formality	The degree to which the problematic situation is presented clearly and precisely.
Requirements fixation	The tendency to rely too heavily on given desiderata when designing a software system.
Critical thinking	"Disciplined thinking that is clear, rational, open-minded, and informed by evidence" [12].
Creativity	"production of novel and useful ideas by an individual or small group of individuals working together" [60].
Software engineering success	Net impact of a system on stakeholders over time [24].

avoid TRS and over-structuring, over-simplifying and over-rationalizing problem statements. Try to present desiderata in ways that encourage skepticism and critical thinking. Information presented to the designers should ideally be less-structured and easily modifiable. In contrast, a clear, well-structured TRS might help in high correctness of code, which can increase the possibility of success of a software developed for mission- or safety-critical domains.

For researchers, it is critical to abandon the naive view that analysts elicit requirements and that design transforms them into appropriate system features. This view obscures the actual relationship between RE and design, which remains contested and poorly understood. SE occurs in complicated situations where stakeholders disagree on system goals and desired features [20]. Analysts and users co-construct evanescent preferences rather than eliciting firm and robust requirements [6]. Design is a creative, improvised, non-deductive process in which designers imagine new systems rather than rearrange old ideas [102]. Expert designers in other fields resist initial problem frames and solution conjectures; they do not deliver requirements in a box-checking manner [39]. Serious questions regarding how best to record and present desiderata remain unanswered. For now; however, we are confident that presenting desiderata in an TRS hinders creativity by inducing fixation and hindering critical thinking.

Making concrete recommendations for SE education is more difficult. Obviously, courses that present an outdated, positivist view of RE should be updated. Non-empirical legacy concepts such as the waterfall model and project triangle should be replaced with evidence-based concepts and theories. Beyond that, we want to recommend teaching a host of underrepresented subjects including design thinking, creativity techniques and theories of cognitive biases. However, SE curricula are already tight. Perhaps a more tractable approach is to transition students to less and less structured assignments as they advance. More open-ended assignments with ambiguous goals, conflicting stakeholder preferences, ill-structured problems and incomplete specifications should help prepare students for more realistic software contexts.

5.3 Quality criteria and threats to validity

We see protocol studies as most consistent with critical realism [103]. Critical realism is a body of philosophical work that attempts to solve Hume's problem of induction by merging a realist ontology with a relativist ontology. Critical realism is fundamentally different from both positivism and constructivism. Positivism (and falsificationism) view reality as observer-independent, objective, measurable and characterized by universal, deterministic, counterfactual, causal laws. Constuctivism, meanwhile, views reality as observer-dependent, subjective and devoid of universal laws because knowledge is context-dependent. Positivism embraces (epistemological) realism; Constructivism embraces epistemological and ontological relativism.

In contrast, critical realism blends a realist ontology ("transcendental realism") with a relativist epistemology ("critical naturalism"). In other words, critical realism assumes that the phenomena that scientists study are *real* whether they can be directly observed (e.g. people, length, Mars) or not (e.g. electrons, creativity, quasars). But because social reality resists experimental closure and is rich in unobservable properties, reality is only imperfectly and "probabilistically apprehensible". Rather than discovering causal laws, scientists therefore construct explanations based on "generative mechanisms"—the powers objects have to influence each other.

Since critical realism is fundamentally different from positivism and constructivism, it has different evaluation criteria, namely—"ontological appropriateness", "contingent validity", multivocality, "trustworthiness", "analytic generalization" and "construct validity" [104]. These criteria do not map neatly into either positivist criteria (internal validity, external validity, etc.) or constructivist criteria (credibility, transferability, etc.)

Critical realism is *ontologically appropriate* because the whole point of a protocol study is to explore cognitive phenomena that are real but cannot be observed directly. *Contingent validity* is the degree to which the study explores generative mechanisms rather than deterministic causal laws. Again, here we are explicitly concerned with exploring the generative mechanism that accounts for design creativity.

Multivocality—the degree to which research integrates diverse perspectives—is typically achieved through data triangulation, which is difficult in a protocol study. Our dialogue–based approach allows us to examine the statements of each half of a participant-pair, as well as directly observing and comparing pairs. However, we cannot corroborate our findings against independent data sources such as archival records, like in a case study. Moreover, brain scans are not yet sophisticated enough to cross-check the inferences we make from participant's verbalizations.

Trustworthiness refers to the chain of evidence from observations to conclusions (see Tables 1 and 2) and the ability of an independent researcher to audit or replicate the findings. We provide as much detail of our analysis process as possible within space limitations, and all of the materials necessary to run an identical study with new participants.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3112503, IEEE Transactions on Software Engineering

However, for privacy reasons, we cannot publish the full transcripts of the design sessions and therefore an independent researcher cannot directly audit our coding.

As we refined the conceptualization of themes, we often renamed or merged multiple themes and their corresponding labels. For example, the theme *expressing values* was renamed to *considering quality criteria* and merged with an earlier theme *non-functional requirements*, and the labels *discussing alternative ideas* was renamed to *discussing design plans*. Despite much refining of labels, the themes and their relationships stabilized early and remained stable. The frequencies of the labels (i.e., *fixation* and *critical thinking*) in Table 2 are unweighted and do not provide evidence of the total number of ideas gained or lost due to one instance of critical thinking and fixation respectively. In other words, one instance of fixation might be more or less important than one instance of critical thinking for creativity.

Moreover, since creativity is only one of the many antecedents of SE project success [24], less constrained and restrictive presentations of desiderata may also undermine success through some other mechanisms, e.g., legal or mission/safety critical constraints. Furthermore, TRS could affect creativity through mechanisms other than those considered in this study.

Analytic generalization refers to generalizing from observations to theory, rather than from a sample to a population. We generalize from observations of designers to the theory shown in Fig. 3.

A protocol study is non-statistical, non-sampling research, using a convenience sample of participants completing a particular task in a particular environment. Our participants were mostly male, non-native English speakers working in English, completing a single, artificial task, in an unfamiliar design domain, in an artificial environment, using artificial task materials, while being watched. All of these factors may have affected our resulted in unknown ways. Results cannot be statistically generalized to different people, other tasks, other environments, or other ways of representing desiderata (e.g. user stories, goal models).

Construct validity refers the degree to which the operationalization and measurement of constructs supports scientific inferences. The only constructs in this study are fixation and critical thinking, which we operationalize by having an expert judge identify them. However, this labeling process is intrinsically subjective, so another analyst might label the data differently. We mitigated this threat by having the second and third authors review the first author's coding, leading to numerous revisions and clarifications. However, a different research team might still produce different labeling.

5.4 Future research directions

We see several promising avenues for future work:

1) Experimentally comparing different representations (e.g., user stories, use cases, code) of the same desiderata to determine which representation is most effective to foster creativity in different circumstances;
2) creating techniques, tools and practices for modelling and managing ambiguity and conflict; and
3) using eye-tracking or protocol analysis to study what professionals attend to and ignore while designing software.

Moreover, requirements fixation is just one of several cognitive biases that may hamper creativity in software design. Future work should investigate related cognitive phenomena including:

1) Confirmation bias—attending disproportionately to information that confirms our current beliefs [72].
2) Miserly information processing—the tendency to avoid deep or complex information processing [85]
3) Conceptual fixation—considering only one or a small number of solution concepts [35].
4) Design fixation—sticking too closely to given or known examples [105].

While confirmation bias, miserly information processing, conceptual fixation and design fixation have all been studied extensively, little work has investigated their effects on software design in particular [106].

6 Conclusion

In summary, desiderata are things that project stakeholders prefer, want or need in a software system. Desiderata can be presented in many ways (e.g. templated requirements specifications, user stories). Previous research showed that presenting desiderata as templated requirements specifications led to less creative designs. We therefore conducted a dialog-based protocol analysis to investigate the cognitive mechanism by which templated specifications affects design creativity. We analyzed the data in two ways: inductive process coding and closed coding. Process coding revealed seven kinds of design actions: making design moves, uncritically accepting, rejecting, grouping, questioning, assuming and considering quality criteria. Closed coding showed that actions associated with requirements fixation are significantly more frequent than actions associated with critical thinking.

These results suggest that presenting desiderata more restrictively as templated requirements specifications is associated with less critical evaluation of task structure and less critical thinking. In other words, templated requirements specifications inhibit design creativity because designers get fixated on desiderata presented (i.e. written) restrictively, well-structured and constrained language, hindering critical thinking.

This paper therefore makes three main contributions: (1) it advances a theory that explains the (previously established) relationship between templated requirements specifications and design creativity; (2) it elaborates the concept of requirements fixation; (3) it presents a simple taxonomy of software design actions. However, our results do not indicate that requirements analysis is useless or that more analysis is counterproductive to creativity. The paper just attempts to present the underlying cognitive mechanisms explaining the effects of presenting desiderata in a very specific way—as templated requirements specifications—on design creativity.

While previous experimental research has demonstrated that presenting desiderata as templated requirements specifications reduces design creativity, our current research explores the underlying cognitive mechanisms that explain this relationship. The results of this study indicate that, given templated requirements specifications, software designers do not proceed as we might hope. Designers should carefully evaluate each desideratum before accepting or rejecting it for articulable reasons. Our observations suggest that designers tend neither to critically evaluate requirements nor to reject questionable ones.

7 Data availability

A comprehensive replication package including all the task documents (i.e., a list of prioritized TRS, demographic questionnaire and blank design template) and the results of the process coding analysis with example quotes are stored in the Zenodo open data archive [107]. (Note: we do not include the transcribed recordings in the replication package to maintain the anonymity of the participants).

Acknowledgment

This study was partially supported by the HPY: Research Foundation (HPY:n Tutkimussäätiö Apurahat) grant.

References

[1] T. Chow and D.-B. Cao, "A survey study of critical success factors in agile software projects," *Journal of Systems and Software*, vol. 81, no. 6, pp. 961–971, 2008.

[2] R. Schmidt, K. Lyytinen, and P. C. Mark Keil, "Identifying software project risks: An international delphi study," *Journal of Management Information Systems*, vol. 17, no. 4, pp. 5–36, 2001.

[3] IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board, "IEEE recommended practice for software requirements specifications," Institute of Electrical and Electronics Engineers, standard, 1998.

[4] J. Mund, D. M. Fernandez, H. Femmer, and J. Eckhardt, "Does quality of requirements specifications matter? combined results of two empirical studies," in *International Symposium on Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE*. Beijing, China: IEEE, 2015, pp. 1–10.

[5] P. Ralph, "The illusion of requirements in software development," *Requirements Engineering*, vol. 18, no. 3, pp. 293–296, 2013.

[6] S. Lichtenstein and P. Slovic, *The construction of preference*. Cambridge, UK: Cambridge University Press, 2006.

[7] P. Rodríguez, E. Mendes, and B. Turhan, "Key stakeholders' value propositions for feature selection in software-intensive products: An industrial case study," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1340–1363, 2018.

[8] T. Sedano, P. Ralph, and C. Péraire, "The product backlog," in *Proceedings of the 41st International Conference on Software Engineering*. Montreal, Canada: IEEE, 2019, pp. 200–211.

[9] R. Mohanani, P. Ralph, and B. Shreeve, "Requirements fixation," in *Proceedings of the 36th International Conference on Software Engineering*. Hyderabad, India: ACM, 2014, pp. 895–906.

[10] R. Mohanani, B. Turhan, and P. Ralph, "Requirements framing affects design creativity," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 936–947, 2021.

[11] P. Ralph, "Toward a theory of debiasing software development," in *EuroSymposium on Systems Analysis and Design*. Gdańsk, Poland: Springer, 2011, pp. 92–105.

[12] Dictionary.com, "Critical thinking," https://www.dictionary.com/browse/critical-thinking, 2019, accessed: 2019-07-10.

[13] F. P. Brooks Jr, *The design of design: Essays from a computer scientist*. Massachusetts, USA: Pearson Education, 2010.

[14] A. T. Bahill and F. F. Dean, "The requirements discovery process," in *INCOSE International Symposium*, vol. 7, no. 1. Wiley Online Library, 1997, pp. 340–347.

[15] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE Transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[16] M. Jackson, "The world and the machine," in *1995 17th International Conference on Software Engineering*. IEEE, 1995, pp. 283–283.

[17] D. L. Parnas and J. Madey, "Functional documents for computer systems," *Science of Computer Programming*, vol. 25, no. 1, pp. 41–61, 1995.

[18] P. Ralph, "The two paradigms of software development research," *Science of Computer Programming*, vol. 156, pp. 68–89, 2018.

[19] J. A. Goguen and M. Jirotka, *Requirements Engineering: Social and Technical Issues*. London, UK: Academic, 1994.

[20] P. Checkland and J. Scholes, "Soft systems methodology: a thirty year retrospective," *Systems Research and Behavioral Science*, vol. 17, no. S1, pp. 11–58, 2000.

[21] B. Nuseibeh, J. Kramer, and A. Finkelstein, "Viewpoints: meaningful relationships are difficult!" in *Proceedings of 25th International Conference on Software Engineering, 2003.* IEEE, 2003, pp. 676–681.

[22] A. Van Lamsweerde, R. Darimont, and E. Letier, "Managing conflicts in goal-driven requirements engineering," *IEEE Transactions on Software Engineering*, vol. 24, no. 11, pp. 908–926, 1998.

[23] L. Goldin and D. M. Berry, "Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation," *Automated Software Engineering*, vol. 4, no. 4, pp. 375–412, 1997.

[24] P. Ralph and P. Kelly, "The dimensions of software engineering success," in *Proceedings of the 36th International Conference on Software Engineering*. Hyderabad, India: ACM, 2014, pp. 24–35.

[25] C. Potts and W. C. Newstetter, "Naturalistic inquiry and requirements engineering: reconciling their theoretical foundations," in *Proceedings of ISRE'97: 3rd IEEE International Symposium on Requirements Engineering*. IEEE, 1997, pp. 118–127.

[26] M. Jackson and P. Zave, "Domain descriptions," in *Proceedings of the IEEE International Symposium on Requirements Engineering*. IEEE, 1993, pp. 56–64.

[27] D. H. Jonassen, "Instructional design models for well-structured and iii-structured problem-solving learning outcomes," *Educational Technology Research and Development*, vol. 45, no. 1, pp. 65–94, 1997.

[28] P. Ralph and R. Mohanani, "Is requirements engineering inherently counterproductive?" in *2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture*. IEEE, 2015, pp. 20–23.

[29] N. Cross, "Expertise in design: an overview," *Design Studies*, vol. 25, no. 5, pp. 427–441, 2004.

[30] J. Wirth, J. Künsting, and D. Leutner, "The impact of goal specificity and goal type on learning outcome and cognitive load," *Computers in Human Behavior*, vol. 25, no. 2, pp. 299–305, 2009.

[31] T. B. Ward, M. J. Patterson, and C. M. Sifonis, "The role of specificity and abstraction in creative idea generation," *Creativity Research Journal*, vol. 16, no. 1, pp. 1–9, 2004.

[32] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991.

[33] R. Guindon, "Knowledge exploited by experts during software system design," *International Journal of Man-Machine Studies*, vol. 33, no. 3, pp. 279–304, 1990.

[34] C. Kruger and N. Cross, "Solution driven versus problem driven design: strategies and outcomes," *Design Studies*, vol. 27, no. 5, pp. 527–548, 2006.

[35] R. J. Youmans and T. Arciszewski, "Design fixation: Classifications and modern methods of prevention," *AI EDAM*, vol. 28, no. 2, pp. 129–137, 2014.

[36] R. A. Finke, "Imagery, creativity, and emergent structure," *Consciousness and Cognition*, vol. 5, no. 3, pp. 381–393, 1996.

[37] O. Akin *et al.*, "Expertise of the architect," *Expert Systems for Engineering Design*, pp. 173–196, 1988.

[38] N. Cross, K. Dorst, and N. Roozenburg, "Research in design thinking," *Proceedings of a Workshop Meeting Held at the Faculty of Industrial Design Engineering*, 1992.

[39] N. Cross, "Design cognition: Results from protocol and other empirical studies of design activity," in *Design Knowing and Learning: Cognition in Design Education*. Oxford, UK: Elsevier, 2001, pp. 79–103.

[40] A. Niknafs and D. Berry, "The impact of domain knowledge on the effectiveness of requirements engineering activities," *Empirical Software Engineering*, vol. 22, no. 1, pp. 80–133, 2017.

[41] G. Mehrotra and D. M. Berry, "How to benefit from newbies' domain ignorance in software development projects," *Science of Computer Programming*, vol. 204, p. 102593, 2021.

[42] N. Maiden, S. Jones, K. Karlsen, R. Neill, K. Zachos, and A. Milne, "Requirements engineering as creative problem solving: A research agenda for idea finding," in *IEEE 18th International Requirements Engineering Conference (RE)*. IEEE, 2010, pp. 57–66.

[43] N. Maiden and A. Gizikis, "Where do requirements come from?" *IEEE Software*, vol. 18, no. 5, pp. 10–12, 2001.

[44] S. Chakraborty, S. Sarker, and S. Sarker, "An exploration into the process of requirements elicitation: A grounded approach," *Journal of the Association for Information Systems*, vol. 11, no. 4, p. 212, 2010.

[45] K. Schmid, "A study on creativity in requirements engineering," *Softwaretechnik-Trends*, vol. 26, no. 1, pp. 20–21, 2006.

[46] B. Crawford, C. L. de la Barra, R. Soto, and E. Monfroy, "Agile software engineering as creative work," in *Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering*. IEEE Press, 2012, pp. 20–26.

[47] N. Maiden, A. Gizikis, and S. Robertson, "Provoking creativity: Imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, pp. 68–75, 2004.

[48] R. Horowitz, "Creative problem solving in engineering design," *PhD. Diss., Tel-Aviv University*, 1999.

[49] L. Nguyen and G. Shanks, "A framework for understanding creativity in requirements engineering," *Information and Software Technology*, vol. 51, no. 3, pp. 655–662, 2009.

[50] J. Guilford, "Three faces of intellect," *Teaching Gifted Students: A Book of Readings*, p. 7, 1965.

[51] M. Michalko, *Thinkertoys: A Candbook of Creative-Thinking Techniques*. Random House Digital, Inc., 2006.

[52] R. B. Svensson and M. Taghavianfar, "Selecting creativity techniques for creative requirements: An evaluation of four techniques using creativity workshops," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 66–75.

[53] N. Maiden, S. Manning, S. Robertson, and J. Greenwood, "Integrating creativity workshops into structured requirements processes," in *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques*. Cambridge, USA: ACM, 2004, pp. 113–122.

[54] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd re," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 176–185.

[55] A. van Lamsweerde, "Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]," in *Proceedings of 12th IEEE International Requirements Engineering Conference, 2004.* IEEE, 2004, pp. 4–7.

[56] J. Horkoff, N. Maiden, and J. Lockerbie, "Creativity and goal modeling for software requirements engineering," in *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 2015, pp. 165–168.

[57] M. A. Boden, *The creative mind: Myths and Mechanisms*. Psychology Press, 2004.

[58] N. Maiden and S. Robertson, "Integrating creativity into requirements processes: Experiences with an air traffic management system," in *13th IEEE International Conference on Requirements Engineering (RE'05)*. IEEE, 2005, pp. 105–114.

[59] T. Bhowmik, N. Niu, A. Mahmoud, and J. Savolainen, "Automated support for combinational creativity in requirements engineering," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 243–252.

[60] T. M. Amabile, "A model of creativity and innovation in organizations," *Research in Organizational Behavior*, vol. 10, no. 1, pp. 123–167, 1988.

[61] R. Mohanani, P. Ram, A. Lasisi, P. Ralph, and B. Turhan, "Perceptions of creativity in software engineering research and practice," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2017, pp. 210–217.

[62] R. J. Sternberg, *Handbook of Creativity*. Cambridge University Press, 1999.

[63] M. Rhodes, "An analysis of creativity," *The Phi Delta Kappan*, vol. 42, no. 7, pp. 305–310, 1961.

[64] J. A. Plucker and M. C. Makel, "Assessment of creativity," *The Cambridge Handbook of Creativity*, pp. 48–73, 2010.

[65] R. E. Mayer, "22 fifty years of creativity research," *Handbook of Creativity*, vol. 449, 1999.

[66] H. H. Christiaans, "Creativity as a design criterion," *Communication Research Journal*, vol. 14, no. 1, pp. 41–54, 2002.

[67] H. G. Nelson and E. Stolterman, *The design way: Intentional change in an unpredictable world: Foundations and fundamentals of design competence*. Educational Technology, 2003.

[68] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 4157, pp. 1124–1131, 1974.

[69] A. Tang and Antony, "Software designers, are you biased?" in *Proceeding of the 6th international workshop on Sharing and Reusing Architectural Knowledge - SHARK '11*. New York, New York, USA: ACM Press, 2011, p. 1.

[70] C. Mair and M. Shepperd, "Human judgement and software metrics," in *Proceeding of the 2nd International Workshop on Emerging Trends in Software Metrics - WETSoM '11*. New York, New York, USA: ACM Press, 2011, p. 81.

[71] I. Salman, B. Turhan, and S. Vegas, "A controlled experiment on time pressure and confirmation bias in functional software testing," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1–35, 2018.

[72] G. Calikli and A. Bener, "Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*. Timisoara, Romania: ACM, 2010, p. 10.

[73] G. J. Browne and V. Ramesh, "Improving information requirements determination: a cognitive perspective," *Information & Management*, vol. 39, no. 8, pp. 625–645, 2002.

[74] N. Chotisarn and N. Prompoon, "Forecasting software damage rate from cognitive bias in software requirements gathering and specification process," in *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*. Yangzhou, Jiangsu, China: IEEE, mar 2013, pp. 951–956.

[75] M. Jorgensen and S. Grimstad, "Software Development Estimation Biases: The Role of Interdependence," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 677–693, may 2012.

[76] A. T. Purcell and J. S. Gero, "Design and other types of fixation," *Design Studies*, vol. 17, no. 4, pp. 363–383, 1996.

[77] M. Perttula and P. Sipilä, "The idea exposure paradigm in design idea generation," *Journal of Engineering Design*, vol. 18, no. 1, pp. 93–102, 2007.

[78] E. G. Chrysikou and R. W. Weisberg, "Following the wrong footsteps: fixation effects of pictorial examples in a design problem-solving task." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 31, no. 5, p. 1134, 2005.

[79] Z. Lujun, "Design fixation and solution quality under exposure to example solution," in *IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE)*, vol. 1. Singapore: IEEE, 2011, pp. 129–132.

[80] C. Toh, S. Miller, and G. Kremer, "Mitigating design fixation effects in engineering design through product dissection activities," in *Design Computing and Cognition'12*. Dordrecht, Netherlands: Springer, 2014, pp. 95–113.

[81] R. J. Youmans, "The effects of physical prototyping and group work on the reduction of design fixation," *Design Studies*, vol. 32, no. 2, pp. 115–138, 2011.

[82] B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging inconsistency in software development," *Computer*, vol. 33, no. 4, pp. 24–29, 2000.

[83] D. Zahner, J. V. Nickerson, B. Tversky, J. E. Corter, and J. Ma, "A fix for fixation? rerepresenting and abstracting as creative processes in the design of information systems," *AI EDAM*, vol. 24, no. 2, pp. 231–244, 2010.

[84] J. Kim and H. Ryu, "A design thinking rationality framework: Framing and solving design problems in early concept generation," *Human–Computer Interaction*, vol. 29, no. 5-6, pp. 516–553, 2014.

[85] K. E. Stanovich, *What intelligence tests miss: The psychology of rational thought*. USA: Yale University Press, 2009.

[86] E. I. Karac, B. Turhan, and N. Juristo, "A controlled experiment with novice developers on the impact of task description gran-

ularity on software quality in test-driven development," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[87] K. A. Ericsson and H. A. Simon, *Protocol analysis: Verbal reports as data*. USA: the MIT Press, 1984.

[88] S. Xu and V. Rajlich, "Dialog-based protocol: an empirical research method for cognitive activities in software engineering," in *Empirical Software Engineering, 2005*. Noosa Heads, Australia: IEEE, 2005, pp. 10–pp.

[89] K. Dorst and J. Dijkhuis, "Comparing paradigms for describing design activity," *Design Studies*, vol. 16, no. 2, pp. 261–274, 1995.

[90] A. Hashem, M. T. Chi, and C. P. Friedman, "Medical errors as a result of specialization," *Journal of Biomedical Informatics*, vol. 36, no. 1-2, pp. 61–69, 2003.

[91] J. Hughes and S. Parkes, "Trends in the use of verbal protocol analysis in software engineering research," *Behaviour & Information Technology*, vol. 22, no. 2, pp. 127–140, 2003.

[92] J. Mingers, "Realizing information systems: critical realism as an underpinning philosophy for information systems," *Information and Organization*, vol. 14, no. 2, pp. 87–103, 2004.

[93] E. D. Canedo and R. P. da Costa, "The use of design thinking in agile software requirements survey: a case study," in *International Conference of Design, User Experience, and Usability*. Springer, 2018, pp. 642–657.

[94] C. Vetterli, W. Brenner, F. Uebernickel, and C. Petrie, "From palaces to yurts: Why requirements engineering needs design thinking," *IEEE Internet Computing*, vol. 17, no. 2, pp. 91–94, 2013.

[95] N. Carroll and I. Richardson, "Aligning healthcare innovation and software requirements through design thinking," in *2016 IEEE/ACM International Workshop on Software Engineering in Healthcare Systems (SEHS)*. IEEE, 2016, pp. 1–7.

[96] P. Ralph, "Fundamentals of software design science," Ph.D. dissertation, University of British Columbia, 2010.

[97] J. Saldaña, *The coding manual for qualitative researchers*. UK: Sage, 2015.

[98] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*. Banff, Canada: IEEE, 2011, pp. 275–284.

[99] M.-L. Chiu, "Design moves in situated design with case-based reasoning," *Design Studies*, vol. 24, no. 1, pp. 1–25, 2003.

[100] L. A. Belady and M. M. Lehman, "A model of large program development," *IBM Systems Journal*, vol. 15, no. 3, pp. 225–252, 1976.

[101] D. M. Berry, "The inevitable pain of software development: Why there is no silver bullet," in *International Workshop On Radical Innovations Of Software and Systems Engineering in the Future*. Springer, 2002, pp. 50–74.

[102] D. A. Schon, *The reflective practitioner: How professionals think in action*. London, UK: Basic books, 1984, vol. 5126.

[103] M. Archer, R. Bhaskar, A. Collier, T. Lawson, and A. Norrie, *Critical realism: Essential readings*. Routledge, 2013.

[104] M. Healy and C. Perry, "Comprehensive criteria to judge validity and reliability of qualitative research within the realism paradigm," *Qualitative market research: An international journal*, vol. 3, no. 3, pp. 118–126, 2000.

[105] R. J. Youmans, "Design fixation in the wild: design environments and their influence on fixation," *The Journal of Creative Behavior*, vol. 45, no. 2, pp. 101–107, 2011.

[106] R. Mohanani, I. Salman, B. Turhan, P. Rodríguez, and P. Ralph, "Cognitive biases in software engineering: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1318–1339, 2020.

[107] R. Mohanani, P. Ralph, B. Turhan, and V. Mandic, "How Templated Requirements Specifications Inhibit Creativity in Software Engineering (Replication Package)," Apr. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4678669

Rahul Mohanani, *PhD (Oulu University), MSc (Lancaster University), B.Eng. (Mumbai University),* is a senior scientist at Fortiss GmbH, Munich and an Adjunct Asst. Professor of Software Engineering at IIIT Delhi. His research, intersecting empirical SE, human aspects and design thinking, has been published in premier venues including the *ACM/IEEE International Conference on Software Engineering* and *IEEE Transactions on Software Engineering*. For more information, please visit http://rahulmohanani.net

Paul Ralph, *PhD (British Columbia), B.Sc. / B.Comm (Memorial),* is an award-winning scientist, author, consultant and Professor of Software Engineering at Dalhousie University. His research intersects empirical software engineering, human-computer interaction and project management. Paul is a member of the *IEEE Transactions on Software Engineering* review board and chair of the *ACM Paper and Peer Review Quality Task Force*. For more information, please visit: https://paulralph.name.

Burak Turhan, *PhD (Boğaziçi University),* is a Professor of Software Engineering at the University of Oulu and an Adjunct Professor (Research) in the Faculty of IT at Monash University. His research focuses on empirical software engineering, software analytics, quality assurance and testing, human factors, and (agile) development processes. He is a Senior Associate Editor of Journal of Systems and Software, an Associate Editor of ACM Transactions on Software Engineering and Methodology and Automated Software Engineering, an Editorial Board Member of Empirical Software Engineering, Information and Software Technology, and Software Quality Journal, and a Senior Member of ACM and IEEE. For more information, please visit: https://turhanb.net.

Vladimir Mandić, *PhD (Oulu University),* is an assistant professor of SE at University of Novi Sad, Serbia. He received his PhD degree in Information Processing Science and SE from the University of Oulu, Finland, and M.Sc.E.E from the University of Novi Sad, Serbia. His areas of interest are software process improvement, empirical software engineering, goal-driven measurement approaches, technical debt and value-based software engineering. He is a member of the IEEE Computer Society.