Requirements Fixation: How Requirements 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 more formally (as "requirements") 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 requirements 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 more formal presentations of desiderata reduce creativity due to *requirements fixation*, a cognitive bias in which designers anchor on explicit requirements, hindering critical thinking. More formally, requirements fixation mediates the negative relationship between the formality of desiderata presentation and design 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 stakeholders (e.g. prospective users) about their needs, wants, preferences, etc.;
- synthesizing stakeholders' opinions into some document(s);
- determining the main features that the product will have; and
- creating mock-ups (diagrams) illustrating the main features and user interfaces.

This raises an obvious question: *What kind of documents are best for synthesizing stakeholders' opinions?* Different areas of research seem to have reached different conclusions.

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]. RE 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. (We will call this the RE perspective.)

- *R. Mohanani is with fortiss GmbH, Germany & Dept of CSE, IIIT Delhi. E-mail: rahul.mohanani@gmail.com, rahul.mohanani@iiitd.ac.in*
- *P. Ralph is with the Faculty of Comp. Sci., Dalhousie Univ., Canada. E-mail: paul@paulralph.name*
- *B. Turhan is with the Faculty of IT, Monash Univ., Australia, and 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*

Contrastingly, 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]);
- 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]. Specifying requirements should therefore lead to designs that satisfy contracts but not users, which is antithetical to user-centred design. (We will call this the "product design" perspective.)

Our previous work showed that presenting stakeholder opinions as "requirements," led to less creative product designs [9], [10]. However, experiments like these are not suitable to explore cognitive mechanisms underlying causal effects, so we know *that* presentation affects creativity but we don't know *how*. This raises the following research question.

> ***Research question:*** *What cognitive mechanisms explain how presenting desiderata as requirements reduces design creativity?*

Here, a *cognitive mechanism* is a psychological phenomenon that corresponds to a mediation relationship in a variance theory. 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 [11]. 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.

Next, we review existing literature (Section 2). Then, we describe our research design, and 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

To understand this study, it is crucial to grasp the linguistic and philosophical differences between the way requirements engineering research and design research conceptualize desiderata.

In the RE perspective, software has requirements, analysts elicit them, and success means fulfilling them. Alternatives to formal requirements statements (e.g. personas, mock-ups, use cases) are just requirements in disguise. If the client says "the system should have an integrated grammar checker", then that is a requirement because being requested by the client is what makes something a requirement. Success means delivering what the client requests. Any diagrammatic or verbal presentation of the "integrated grammar checker" feature is a requirements representation.

In the product design perspective, software does not have requirements [5], [12]. Stakeholders have unstable, unreliable, conflicting desiderata [6], [13]. When a stakeholder says "I want the system to have an integrate grammar checker" we know that is a desideratum because everything stakeholders request is a desideratum. However, we do not know if it is a requirement because success means delivering benefits to stakeholders [14] and we do not know if the system actually needs a grammar checker to deliver its intended benefits. Desiderata can be presented as formal requirements statements, or as personas, use cases or user stories. User stories, for example, are not requirements but a different way of working (cf. [8]).

In our experience, many researchers struggle to grasp this distinction because they see their perspective as objective reality rather than *their perspective*. This paper adopts the perspective that stakeholders have desiderata and requirements statements are one of many ways to present desiderata. Only by (temporarily) accepting this perspective can the reader appreciate this paper.

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" [15, p. 65]. A body of empirical research shows that task structure is negatively associated with design performance (cf. [16] for summary). "Over-concentration (over-structuring) on problem definition does not necessarily lead to successful design outcomes" [17, p. 439] for at least five reasons:

1) less specific goals reduce cognitive load [18], which leads to more learning;
2) less specific task framing results in more creative solutions [19];
3) designers often fixate on experience [20] or on initial set of ideas [21]; and
4) designers often process whatever little information they have and quickly assimilate it into the problem schema, improving their understanding of the problem [22].

Perhaps unsurprisingly, then, our recent experiments showed that presenting desiderata as "requirements" reduced creativity [9], [10]. We hypothesized that presenting desiderata as "requirements" triggers a specific cognitive bias (that is, a systematic deviation from optimal reasoning), which they 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" [23, p. 5]. Requirements fixation, then, is the tendency to attribute undue confidence and importance to desiderata presented as formal requirements statements.

Although, the precise mechanism by which increasing task structure reduces design performance in terms of creative solutions remains unclear, design expertise seem to moderate the relationship. Expert designers tend to resist any initial problem framing (e.g., a given list of formal requirements specifications) and proceed via an improvised, solution-focused approach [24]. Expert designers consider all problems as ambiguous and ill-structured, focusing on solution-generation rather than analysing the given problem [17], [25]. On the other hand, novice designers usually end up treating ill-structured problems as well-structured, thereby compromising the potential for any creative solutions [26].

While, the underlying cognitive mechanisms that reduces creativity of design concepts due to specifications presented as formal requirements are not yet well explored, recent experiments in SE (e.g. [9], [10]) have demonstrated the tendency of designers to fixate on desiderata framed as "requirements", which reduces their creativity.

2.3 Creativity

Research in SE considers requirements engineering (RE) as a highly creative process [27], [28], where analysts and multiple stakeholders collaborate together to make sense of the current (problem) situation and conceptualize a common mental-model of a possible system [29]. In RE, creativity enhancing workshops (e.g., [30]) are extensively used to provide clarity for requirements identification [31], [32] and

generate novel and creative requirements [33], [34]. In these workshops, creativity is often linked with divergent thinking [35], i.e., exploring multiple and diverse solutions to a given problem. Moreover, interactive collaboration techniques employed during these workshops help generating more creative requirements [33]. In a nutshell, requirements can be understood as entities that encapsulates the results of creative thinking about the system being developed [36].

Creativity research can be largely categorised into the "Six P's" (viz., [37], [38]), 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*.

In the subsequent study, we analyse the thoughts, actions and design moves employed by the participants while creating conceptual designs of a mobile application (see section 3.4). Hence, we consider creativity in terms of a creative *product*. Creativity, itself, is a poorly understood [39] multi-dimensional construct [40], where an artifact (or *product*) is considered creative in terms of not only novelty (i.e., originality) [41], [42] but also its practicality (i.e., usefulness) [43], [44].

2.4 Fixation

Cognitive biases are systematic deviations from optimal reasoning [45]. Since software engineering involves lots of reasoning, cognitive biases help to explain common problems in software design [46], [47], testing [48], [49], requirements engineering [50], [51] and project management [52].

Fixation is a cognitive bias in which "blind adherence to a set of ideas or concepts limit[s] the output of conceptual designs" [20]. Several experiments have demonstrated *design fixation*—the tendency for designers to generate solutions very similar to given examples [20], [53] or existing artifacts [54]. The cognitive mechanisms underlying design fixation are not well understood. However, Youmans and Arciszewski suggest that designers may fixate on a known but limited set of ideas or an existing body of knowledge. They 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 [55];
2) examples—common examples causes more fixation than unusual or rare examples [56];
3) "defixating" instructions—explicit instructions to avoid features of given or existing artifacts [57];
4) providing good quality examples than flawed or no examples lead to better design performance [58];
5) product dissection activities [59]; and
6) physical prototyping [60].

SE research shows that inconsistency in requirements specifications may reduce premature commitment [61]. More generally, the way a task is communicated or presented may also affect fixation [62], [63]. This is related to the *framing effect*: "the tendency to give different responses to problems that have surface dissimilarities but are formally identical" [64, p. 88].

Desiderata can be presented in many different forms, including 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], [65].

3 Research Design

This section describes our methodology. Fig. 1 summarizes the research protocol.

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 (mental processes) of the subject's working memory" [66].

However, verbalizing our thought process probably changes those thought processes in imperceptible ways. We can mitigate this limitation using *dialog-based protocol analysis* [67], 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 (cf. [68]), psychology (e.g. [69]), medicine (e.g. [70]) and software engineering (cf. [67], [71]).

3.2 Purpose and scenario

The purpose of this study is to determine why presenting desiderata as "requirements" reduces design creativity. To do so, we observe participants in a simulation of a situation where a software team is given a requirements specification and then asked to design an appropriate application. Although requirements engineering (RE) and high-level designing are increasingly merged (e.g. [72]–[74]), 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.

Note: an alternative question—whether modeling desiderata as requirements harms creativity in teams that both create the requirements specification and then design

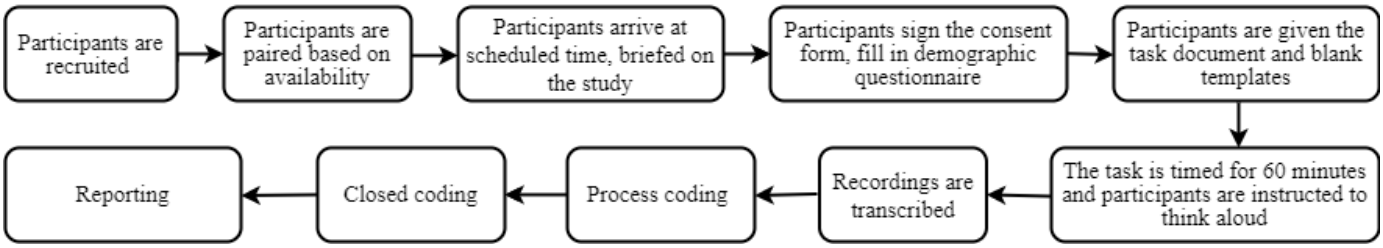

Fig. 1. Overview of the data collection process

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 [75, 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 with the fourth author. 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

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 (including the recording) and participants signed a consent form and completed a demographic questionnaire.

We used the same task as our previous experiments [10]. The task document listed 25 desiderata presented as requirements and organized into five priority levels (high, high-medium, medium, medium-low and low). Crucially, many of the desiderata presented in this task are ill-considered, inconsistent, over-complicated or otherwise dubious. The specification was written 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.

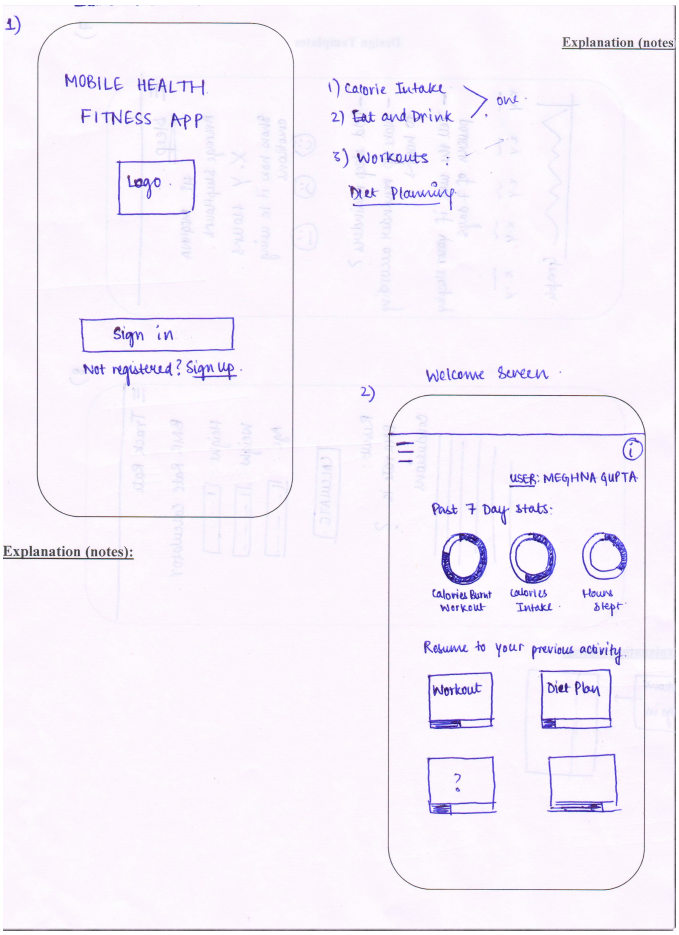

Fig. 2. Example of a conceptual design

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. Fig. 2 is an example of a conceptual design created by participants.

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 are available in our replication package (see Section 7).

TABLE 1
Summary of process coding analysis (Part A)

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.

3.5.1 Process coding

We began by analyzing the data using inductive process coding [76]. That is, we coded (i.e. labeled) 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 altogether. 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 [77]. 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

Next, 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. Tables 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 [78]. *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. Some participants only tried to meet the requirements prioritized 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. Sometimes participants would temporarily ignore a high-priority requirement, other times they 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. Some 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 high-priority requirements while ignoring low-priority requirements is related to participants' uncritical acceptance of priority levels and task framing 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. Some participants backtracked on their earlier design moves based on their evolving understanding of the task.

4.1.6 Assuming

Eleven pairs made explicit *assumptions* about the task. These assumptions were mostly 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. Few 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 some other groups would over-estimate the effort required by creating self-imposed time-constraints. Participants sometimes 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 (i.e. speed), stability, and aesthetics.

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**

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 (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 (14 instances); only two pairs expressed overall doubts about the requirements (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 (Indepen-

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

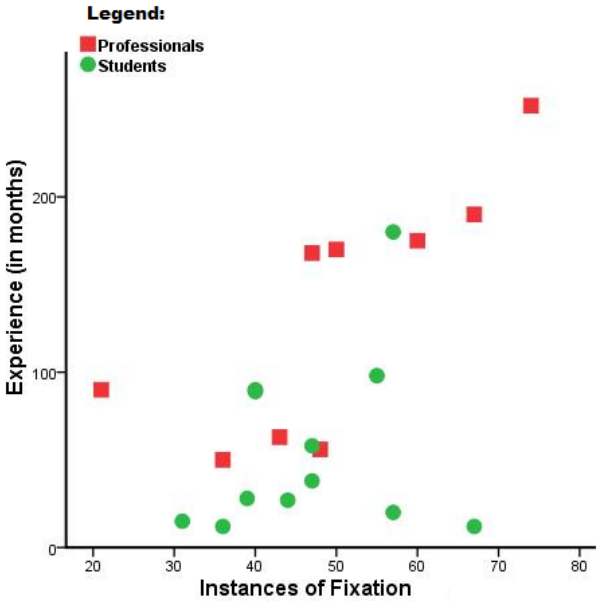

Fig. 3. Scatterplot of experience (months) vs. fixation (instances)

dent 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. 3—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

Our working theory of requirements fixation is shown in Fig. 4 and Table 4. Briefly, one (of many) important antecedents of success is creativity. Previous work [9], [10], [19] showed that presenting specifications more formally

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

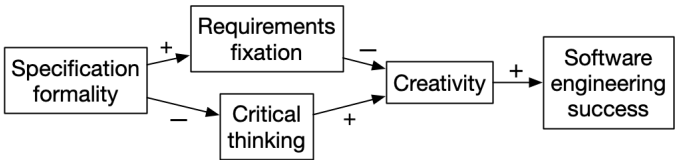

Fig. 4. Theoretical framework

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

(as requirements) diminished creativity, which undermines success. The simulation reported above shows that participants given formal specifications have many more instances of fixation than critical thinking. This suggests that both fixation and critical thinking mediate the relationship between specification formality and creativity. In other words, we believe the nexus of requirements fixation and critical thinking is the cognitive mechanism by which formally specifying requirements hinders creativity.

Furthermore, we identified several indicative behaviours for both fixation and critical thinking—the labels in Table 2. For example, unthinking acceptance of requirements statements indicates requirements fixation; questioning the reasonableness of priority levels indicates critical thinking.

Seen through a different theoretical lens, designing involves rapidly shifting between divergent thinking and convergent thinking [80]. *Divergent thinking* involves generating multiple novel ideas and departing (or *diverging*) from the status quo. Convergent thinking involves evaluating ideas and eventually *converging* on a single course of action. Good design necessitates a reasonable balance of convergent and divergent thinking. Too much convergent thinking is ineffective; too much divergent thinking is inefficient. Participants in our simulation display significantly more convergent thinking than divergent thinking. If specification formally promotes convergent thinking, this explains why presenting desiderata as requirements hinders creativity.

5.2 Implications

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

For professionals, the message is simple. If more creative solutions are desired, do not use a formal requirements specification. Present desiderata in less formally, in ways the promote skepticism. While the spectrum of formality is not well-defined, a traditional software requirements specification is obviously more formal than a collection of sticky notes; a prioritized list of desiderata is more structured than an unordered list. In contrast, a clear, well-structured and a formal list of specifications might help in high correctness of code, which can increase the possibility of success of a software developed for mission- or safety-critical domains.

Professionals may also benefit from techniques for promoting critical thinking and mitigating fixation such as:

1) Avoid labelling desiderata in ways that overstate confidence. Something that might be a good idea is

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" [79].
Creativity	"production of novel and useful ideas by an individual or small group of individuals working together" [40].
Software engineering success	Net impact of a system on stakeholders over time [14].

not an 'optional requirement'; it is an 'unvalidated idea'.

2) Avoid labelling desiderata in ways that overstate their importance. If something is not very important, call it a 'possible future feature' rather than a 'low-priority requirement'.
3) Resist the urge to define 'the problem'. If stakeholders do not agree on a single problem the system should solve, writing a singular problem statement does not foster agreement; it marginalizes dissent. Model disagreements and conflicts instead.
4) Actively manage the product backlog [8]. As the team learns more about the project context, some backlog items should be revealed as undesirable and removed.
5) Run a creativity workshop [36]. Specifically ask for outside-the-box ideas. Try lateral thinking exercises. [81].

For researchers, it is critical to abandon the naive view that analysts elicit requirements and that design transforms them into appropriate system features. SE occurs in complicated situations where stakeholders disagree on system goals and desired features [82]. Analysts and users co-construct evanescent preferences rather than eliciting firm requirements [6]. Design is a creative, improvised, non-deductive process in which designers imagine new systems rather than rearrange old ideas [83]. Expert designers in other fields resist initial problem frames and solution conjectures; they do not deliver requirements in a box-checking manner [26]. Serious questions regarding how best to record and present desiderata remain unanswered. For now; however, we are confident that presenting desiderata as a formal, prioritized requirements specification hinders creativity by inducing fixation and obstructing critical thinking.

Making concrete recommendations for SE education is more difficult. Obviously, outdated requirements engineering courses should be updated to include user-centred design, writing user stories and managing a product backlog. 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. When students are learning how to write for loops, and their instructors want to use automated grading, assignments have to be structured. By final year, however, a perfectly reasonable project specification would be "build something useful and get people to use it." More open-ended assignments with ambiguous goals, conflicting stakeholders preferences, ill-structured problems and incomplete specifications should help prepare students for more realistic software contexts.

5.3 Quality criteria and threats to validity

Protocol analysis is a critical realist qualitative inquiry; therefore, appropriate quality criteria include dimensions such as credibility, resonance, usefulness and transferability rather than positivist criteria like internal validity and generalizability. We establish credibility by illustrating the chain of evidence from observation to theoretical categories (Table 1). We attempted to establish resonance through member-checking (i.e. we shared our results with the participants) but none of the participants provided any substantive feedback. We establish usefulness in Section 5.2 by describing practical implications and recommendations. Transferability is more challenging: in a sense, our concepts apply to any situation where software designers are given specifications; however, that does not mean specification formality hinders creativity in all circumstances.

Protocol studies are widely used in design research because they are the most effective method currently available for examining designers' cognition [71]. However, they have several well-known limitations:

1) Asking participants to think aloud may alter their cognition imperceptibly. Dialog-based protocol studies reduce this threat.
2) Participants may behave unnaturally because they're being observed (the Hawthorne effect [84]).
3) A protocol study is non-statistical, non-sampling research, using a particular task in a particular environment. Results cannot be statistically generalized to a population, or to other tasks or environments.
4) We infer unobservable cognitive processes from verbalizations. Brain scans are not yet sophisticated enough to cross-check these inferences.
5) Qualitative data analysis is inherently subjective. We mitigated this threat by having the second and third authors review the first author's coding, leading to numerous revisions and clarifications.

In our study, some pairs seemed to communicate poorly, and some participants seemed reluctant to think aloud in English. We tried to mitigate this issue by periodically prompting silent pairs to 'please speak out loud', but both their lack of communication and our prompting may have affected their flow of thoughts. That said, hour-long observations of 42 participants produce a rich data set, and our themes saturated well. 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.

More broadly, since creativity is not the only antecedent of success, less formal specifications may undermine success through some other mechanism. Furthermore, specification formality could affect creativity through mechanisms other than those considered in this study.

5.4 Future research directions

We see several promising angles for future work:

1) experimentally comparing different representations (e.g. user stories, use cases, requirements statements) of the same desiderata to determine which representation is most effective 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 the confirms our current beliefs [49].
2) Miserly information processing: the tendency to avoid deep or complex information processing [64]
3) Conceptual fixation: considering only one or a small number of solution concepts [53].
4) Design fixation: sticking too closely to given or known examples [85].

While confirmation bias, miserly information process, conceptual fixation and design fixation have all been studied extensively, little work has investigated their effects on software design in particular [86].

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. requirements statements, user stories). Previous research showed that presenting desiderata more formally (as requirements statements) led to less creative designs. We therefore conducted a dialog-based protocol analysis to investigate the cognitive mechanism by which specification formality 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 formal presentations of desiderata are associated with less critical evaluation of task structure, less critical thinking and less divergent thinking. In other words, **requirements specifications reduce design creativity because designers get fixated on the specifications**.

This paper therefore makes three main contributions: (1) It advances a theory that explains the relationship between specification formality and design creativity; (2) It elaborates the concept of requirements fixation; (3) It presents a simple taxonomy of software design actions.

While previous experimental research has demonstrated that presenting desiderata as formal requirements reduces design creativity, our research explores the underlying cognitive mechanisms that explain this relationship. The results of this study indicate that, given formal requirements statements, 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 requirements specifications, 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 [87]. (Note: we do not include the transcribed recordings in our 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 *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*. 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*, pp. 1–1, 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*, in press.

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

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

[13] P. Checkland and J. Scholes, *Soft systems methodology: a 30-year retrospective*. John Wiley Chichester, 1999.

[14] 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.

[15] 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.

[16] P. Ralph and R. Mohanani, "Is requirements engineering inherently counterproductive?" in *Proceedings of the Fifth International Workshop on Twin Peaks of Requirements and Architecture*. Vienna, Italy: IEEE, 2015, pp. 20–23.

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

[18] 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.

[19] 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.

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

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

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

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

[24] O. Akin *et al.*, "Expertise of the architect," *Expert systems for engineering design*, pp. 173–196, 1988.

[25] 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.

[26] 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.

[27] 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 *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010, pp. 57–66.

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

[29] 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.

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

[31] 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.

[32] 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.

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

[34] 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.

[35] J. Guilford, "Three faces of intellect1," *Teaching Gifted Students: A Book of Readings*, p. 7, 1965.

[36] 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.

[37] R. J. Sternberg, *Handbook of creativity*. Cambridge University Press, 1999.

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

[39] R. Mohanani, P. Ram, A. Lasisi, P. Ralph, and B. Turhan, "Perceptions of creativity in software engineering research and practice," in *Software Engineering and Advanced Applications (SEAA)*. Vienna, Austria: IEEE, 2017.

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

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

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

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

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

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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

[51] 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.

[52] 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.

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

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

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

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

[65] E. I. Karac, B. Turhan, and N. Juristo, "A controlled experiment with novice developers on the impact of task description granularity on software quality in test-driven development," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

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

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

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

[69] K. Ericson and H. Simon, "Protocol analysis: Verbal reports as data. a bredford book," 1984.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

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

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

[77] 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.

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

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

[80] G. Goldschmidt, "Linkographic evidence for concurrent divergent and convergent thinking in creative design," *Creativity research journal*, vol. 28, no. 2, pp. 115–122, 2016.

[81] E. De Bono, *Lateral thinking: a textbook of creativity*. London, UK: Penguin UK, 2010.

[82] P. Checkland, "Soft systems methodology: a thirty year retrospective," *Systems research and behavioral science*, vol. 17, no. S1, pp. 11–58, 2000.

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

[84] J. G. Adair, "The hawthorne effect: a reconsideration of the methodological artifact." *Journal of applied psychology*, vol. 69, no. 2, p. 334, 1984.

[85] 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.

[86] 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*, in press.

[87] R. Mohanani, P. Ralph, B. Turhan, and V. Mandic, "Requirements fixation: How requirements inhibit creativity in software engineering? replication package," 2020. [Online]. Available: 10.5281/zenodo.4006445

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 an Associate Professor in the Faculty of Information Technology at Monash University and an Adjunct Professor at the Univerity of Oulu. His research focuses on empirical software engineering, software analytics, quality assurance and testing, human factors, and (agile) development processes. He is on the editorial/review boards of several journals including *IEEE Transactions on Software Engineering*, *Empirical Software Engineering*, *Journal of Systems and Software*, *Information and Software Technology* and *Software Quality Journal*. He is a senior member of the IEEE, a member of the ACM, ACM SIGSOFT, and IEEE Computer Society. For more information please visit: https://turhanb.net.

Vladimir Mandić, *PhD (Oulu)*, 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 memebr of the IEEE Computer Society.