The Journal of Systems & Software 144 (2018) 22–40

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Improving software design reasoning–A reminder card approach[☆]

Antony Tang[a,*], Floris Bex[b], Courtney Schriek[c], Jan Martijn E.M. van der Werf[b]

[a] *Swinburne University of Technology, Melbourne, Australia*
[b] *Department of Information and Computing Science, Utrecht University, Utrecht, The Netherlands*
[c] *Deloitte, Utrecht, The Netherlands*

A B S T R A C T

Software designers have been known to think naturalistically. This means that there may be inadequate rational thinking during software design. In the past two decades, many research works suggested that designers need to produce design rationale. However, design rationale can be produced to retrofit naturalistic decisions, which means that design decisions may still not be well reasoned. Through a controlled experiment, we studied design reasoning and design rationale by asking participants to carry out a group design. As treatment, we provided 6 out of 12 student teams with a set of reasoning reminder cards to see how they compare with teams without the reminder cards. Additionally, we performed the same experiment with 2 teams of professionals who used the reminder cards, and compared the results with 3 teams of professionals. The experimental results show that both professionals and students who were equipped with the reasoning reminder cards reasoned more with their design. Second, the more a team discusses design reasoning, the more design rationale they find.

1. Introduction

Software design is typically a group activity where many people participate in discussions, exchange ideas and make decisions. In a software team, members can have different levels of technical and domain experience, personalities, biases and relationships with each other. The group collectively makes decisions based on some rationale that is sometimes tacit. The reasoning and argumentation behind design decisions may not be explicitly stated. It has been shown that decisions can be performed in a naturalistic manner without conscious awareness of reasoning (Zannier et al., 2007).

In the past decades, software engineering researchers noticed that design reasoning is a key process in design. In the 90′s, issue-driven design methods such as gIBIS and QOC were proposed to guide designers to focus on design issues and criteria to judge design decisions. From 2000 onwards, many models and techniques were proposed to make use of design rationale and knowledge management in software decision making (Burge and Brown, 2000; Capilla et al., 2016; Falessi et al., 2013). Much of this research focuses on the means to improve the design process by providing a process or a software tool. One of the problems of such methods and tools is the cognitive overload that results from having to learn and use such tools at the same time as having to discuss and think about complex software designs (Buckingham Shum and Hammond, 1994). Furthermore, not much of the research on design reasoning has empirically investigated which techniques and how much information designers use in design reasoning, which information is missing, and how the use of design reasoning techniques affects design discussions and eventually the quality of design discourse.

Based on our previous work, there are reasons to believe that it is advantageous to use *design reasoning techniques* – techniques that can be applied to elicit information and reasoning about a design – during design discussions (Razavian et al., 2016). Conversely, incomplete and implicit design information can limit the communication and understanding of design issues between team members and hamper design argumentation (Børte et al., 2012). In a study, it was found that by asking novice designers to verbalize their options and their justifications, the quality of a design improved (Tang et al., 2008). If we can find a way to help designers, novice as well as professionals, to explicitly carry out design reasoning techniques it may help us better tailor software engineering methods to assist designers. The focus of such an approach would be to provide a reminder method to help designers think better during the process of design.

The aim of this research is to investigate whether a simple, reflective method based on a set of reasoning reminder cards could actively trigger the use of design reasoning techniques and verbalize explicitly

[☆] This journal article is an extension of our work based on C. Schriek, J. M. E. M. van der Werf, A. Tang, and F. Bex, "software architecture design reasoning: a card game to help novice designers," in *software architecture: 10th European conference, ECSA 2016.*
[*] Corresponding author.
E-mail address: atang@swin.edu.au (A. Tang).

https://doi.org/10.1016/j.jss.2018.05.019
Received 14 April 2017; Received in revised form 21 February 2018; Accepted 17 May 2018
Available online 18 May 2018
0164-1212/ © 2018 Elsevier Inc. All rights reserved.

more design rationale, that is, the design issues, design decisions and rationales behind these decisions. Part of the research is purely descriptive: which techniques and information do designers use in design reasoning? We also have a normative goal: the reasoning reminder card approach prescribes a (lightweight) method and suggests designers to use design reasoning techniques. We test if such an approach would improve reflective thinking during the design discourse without putting an extra cognitive burden on the designers. Consistent with our previous research works (Tang and Lau, 2014; Schriek et al., 2016), the goal is to enable the creation of design methods to facilitate design reasoning and the generation of design rationale *during* design discourse. We differentiate our approach with the traditional software architecture inspection and evaluation approaches in that our approach focuses on reasoning during design discourse, whereas the traditional architecture evaluation approach evaluates the completed design, typically *after* most of the design discourse has completed and design ideas may have been well anchored.

Our starting point is to design a set of reasoning reminder cards, or reminder cards in short, based on the design reasoning process, techniques and rationales that are documented in the literature (Razavian et al., 2016; Poort and v. Vliet, 2011; Kazman et al., 1998; Dorst and Cross, 2001). We have created seven cards, where each card represents an aspect of design reasoning. The card system is visual and simpler than the method reported in (Razavian et al., 2016). We conjecture that it can remind the participants to reason more. We observed 12 teams of students and 5 teams of professionals solving the same design problem (Petre and Van Der Hoek, 2013). The 17 teams were divided into two groups: the treatment or test group consisting of teams who used the reasoning reminder cards, and a control group, being the design teams without any cards. We postulate that the reasoning reminder card approach could help design teams in the test group to apply more design reasoning techniques than the teams in the control group, which allows them to communicate more explicitly the design rationale in a design discourse.

The research results presented in this paper extend our previous research paper (Schriek et al., 2016). In our previous work, we found that students who used the reminder cards used more reasoning techniques and found more design rationale. In this extended work, we have added a number of new findings. Firstly, we added five teams of professionals and investigated how professionals and students differ when using the card system. Secondly, we investigated in more depth the relationships between card play and the identification of design rationale. Thirdly, we analysed how the use of reminder cards encouraged designers to engage in more discussions.

In this extended research, we found that the teams in the test group, both students and professionals, used more reasoning techniques and found more design rationale with the aid of the reminder card approach. This result shows that the reminder cards have impacts on design discourse and the design outcomes. Teams that were equipped with the reasoning reminder cards reasoned significantly more than teams that use a naturalistic approach. It appears that the reminder cards have provided a structure with which designers can use to generate and reason with design ideas. As such, more design discussion was observed and more design rationale was found. We have also shown in this research a relationship between reasoning and design rationale, the more reasoning is discussed, the more design rationale is discovered.

The remainder of the paper is structured as follows. The next section discusses design reasoning and rationale, explaining the current state of the art in literature. Section 3 describes the research approach we followed to design the reasoning reminder cards, and how these cards were used as treatment in the controlled experiment we performed. The results of the experiment are presented in Section 4, and the results are discussed and analysed in Section 5. Possible threats to validity are discussed in Section 6. Last, Section 7 concludes the paper.

2. Software decision making related work

Software design is described as wicked by many (Vliet, 2008; Dutoit et al., 2006). One of the issues is wicked problem characterization (Rittel and Webber, 1973). Unknown issues in the problem space and the solution space create complexities in software decision making. During design, there is no definite formulation to solve a design problem. There is no stopping rule to tell when an acceptable solution is being reached. It is difficult to tell if a design truly works or not, at least not before implementation. Resolving one design issue can give rise to other related design issues (van Vliet and Tang, 2016). These phenomena make the act of software design complex.

Adding to the complexity of software design, humans do not always make rational and objective decisions because of many decision-making limitations. One of these limitations is cognitive bias. A cognitive bias is a systematic distortion of human ability to reason (Kahneman and Tversky, 1972). This issue has been shown to have affected economic reasoning (Kahneman, 2003; Tversky and Kahneman, 1986). If the cognitive biases that have been identified in everyday economic activities also exist in software design activities, then this is problematic for the software engineering discipline. We need to also ask how we may detect, prevent and correct these biases. Another decision-making limitation is satisficing, Tang and Vliet (Tang and van Vliet, 2015) showed in a study that designers, both students and professionals, do not reason much before they decide. Designers tend to be satisfied with a good enough solution without thorough reasoning. Much of the decision making is based on gut feeling, or naturalistic decision making (Zannier et al., 2007). The possible result is sub-standard decisions.

2.1. Design reasoning processes, techniques and rationales

To combat the above-mentioned “wickedness” and the tendency towards common reasoning biases in the software design process, many methods and techniques have been proposed. Based on the seminal IBIS work from the late 70′s, methods such as gIBIS and QOC have been proposed to focus designers on the central issues, questions, and rationales (Conklin and Begeman, 1988; Maclean et al., 1996; Conklin et al., 2001; Buckingham Shum et al., 2006). These methods are centered around *issues* in the design, for which *options* (or positions) can be proposed. These options can be sup ported or attacked by arguments *pro* and *con*, respectively. Furthermore, it is suggested to identify individual *constraints* on the design, *risks* involved with the design, and *assumptions* underlying the design.

From 2000 onwards, many more models and techniques were proposed to make use of design rationale in software decision making, such as risk analysis, trade-offs, assumption analysis and problem structuring (see (Capilla et al., 2016) for a brief overview). *Problem structuring*, for example, is used to identify the problem space by asking questions related to the problem, such as what are the key issues. *Option generation* is a technique specifically directed at the problem of anchoring by designers: the designer is forced to consider the different available options to solve a design problem (Dorst, 2006; Tang et al., 2010). Both problems and solution options co-evolve as the problem and solution spaces are explored by designers (Dorst and Cross, 2001; Wiltschnig et al., 2013). *Constraint analysis* looks at the constraints exerted by the requirements, context and earlier design decisions and how they impact the design (Tang et al., 2010; Zimmermann et al., 2009). *Risk analysis* is a technique to identify any risks or unknowns and their impact that adversely affect the design (Poort and v. Vliet, 2011; Boehm, 1991). *Assumption analysis* is a technique to question the often implicit assumptions underlying other elements of the design discourse, such as issues and constraints (Lago and van Vliet, 2005). Finally, *trade-off analysis* is a technique to help assess and make compromises when design issues or design options conflict. It can be used for prioritization of problems and to weigh the pros and cons of a design (Kazman et al., 1998; Bass et al., 2012). We call these techniques *design reasoning*

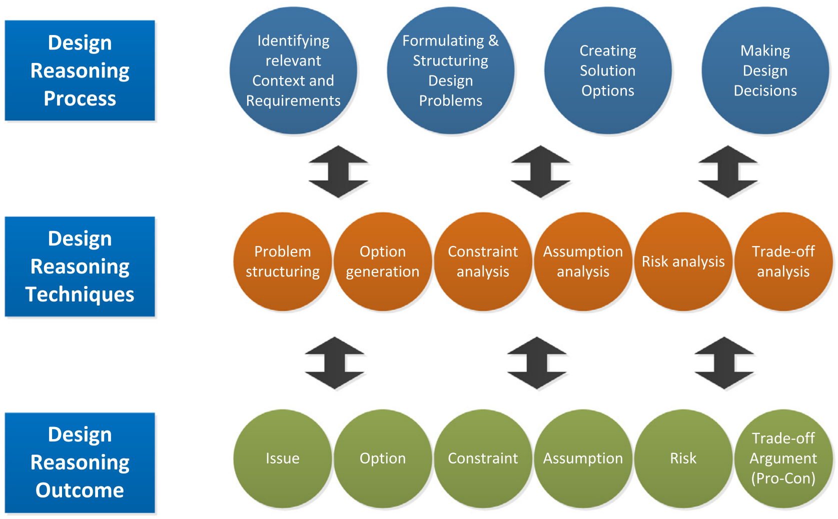


Fig. 1. Interaction between design reasoning process, techniques and rationales.

techniques. Design reasoning using these techniques differ from design rationale models in that they emphasize on the reasoning process instead of the outcomes (i.e. design rationale).

On the basis of Design Association Theory proposed in (Tang and Lau, 2014), (Razavian et al., 2016) proposed a general model for reflective design reasoning process, consisting of *identifying context and requirements, formulating and structuring design problems, creating solution options* and *making design decisions*. The contexts and requirements help define the design problems. The identification of design problems helps designers to frame potential solution options. The solution options are evaluated through trade-off analysis to decide on the most appropriate solution. Subsequent contexts and problems are created as partial solutions.

As can be seen in the above literature overview, there are many models and reasoning techniques to help generate design rationale. Fig. 1 shows a conceptual model summarising the three aspects of design reasoning:

1. Design reasoning process: the general stages of reasoning in a design process (Tang and Lau, 2014).
2. Design reasoning techniques: the specific reasoning techniques and methods applied in the design reasoning process (Tang et al., 2010).
3. Design reasoning outcomes: the outcomes of the discussions in which reasoning techniques have been applied; the explicit documentation of the issues, decisions and rationales behind these decisions (Buckingham Shum et al., 2006; Tyree and Akerman, 2005).

Note that we make a clear distinction between the design reasoning process and the techniques applied in this process on the one hand, and the design reasoning outcomes, the end product, on the other hand. For example, during the reasoning process designers perform assumption analyses (design reasoning technique), which produces the various assumptions (design reasoning outcomes) underlying the design. A conceptual model such as the one presented in Fig. 1 is a useful tool in depicting the elements of design reasoning. In itself, it does not help designers to reason systematically with a design, nor could it overcome issues such as cognitive biases, satisficing and cognitive limitations. However, it provides a framework in which methods can be created to help designers to reason with a design. These methods are discussed below.

2.2. Software architecture evaluation

Traditional approach to evaluating software design is typically having a separate evaluation team to assess an end-design or software. It was found that software inspection and evaluation benefit software quality (Kollanus and Koskinen, 2009). There are many such techniques. Examples of software architecture evaluation techniques are: Scenario-Based Analysis of Software Architecture (SAAM) (Kazman et al., 1996), Architecture Tradeoff Analysis Method (ATAM) (Bass et al., 2012), SARA Report (Obbink et al., 2002), Lightweight Architecture Alternative Assessment Method (LAAAM) (Carriere, 2005), Tiny Architectural Review Approach (TARA) (Woods, 2012), Scenario-Based Architecture Reengineering (SBAR) (Bengtsson and Bosch, 1998) and Cost Benefit Analysis Method (CBAM) (Kazman et al., 2001). Many of these works proposed to review inputs and outcomes, and some of them provide a checklist for assessments. They generally assume that a design is available for evaluation and reason with a design outcome. However, some of them do not improve design reasoning and design thinking during design construction. Checklists and scenarios can also be useful (Abowd et al., 1997). They can prompt designers to consider the specific situations but they are also context dependent. We do not know if they improve design reasoning.

The difference between evaluating a design and evaluating a design discourse is that the latter is about self-reflection and reasoning, and the focus is on the design discourse. Designers are steered to be cognisant of the reason they put forward when they consider a design. If a design is reasoned carefully and thoughtfully, we suggest that many of the design errors can be avoided before the software architecture evaluation stage.

2.3. Reflection and reasoning reminders

It is recognised that cognitive biases can influence the judgment of a designer (Stacy and MacMillan, 1995; Zalewski et al., 2017). For instance, a software designer can fixate or anchor on a solution that first came to his mind and as a result ignores other options. One way to counter such a cognitive bias is by way of reflection and challenge (Reymen, 2001). Lockyer et al. adopted a model of reflection in which reflection comprises external feedback and self-distillation (Lockyer et al., 2004). Schön pointed out that a designer must consider the situations that he or she is faced with, and reflect on it to obtain new ideas (Schön, 1983). Reflection can be considered as a conversation about the thinking process. Reflection is close to the agile philosophy and can be treated as practices integrated into the agile software construction process (Babb et al., 2014). Reflection in design sessions (Reymen, 2001), in software development (Babb et al., 2014; Bull and Whittle, 2014) and in agile development (Schwaber and Beedle, 2002;

Table 1
Reflective questions (adapted from (Razavian et al., 2016)).

	Design contexts and requirements	Design problems	Design solutions
Assumption analysis	What assumptions are made?	Do the assumptions affect the design problem?	Do the assumptions affect the solution option?
Risk analysis	What are the risks that certain events would happen?	How do the risks cause design problems?	How do the risks affect the viability of a solution?
Constraint analysis	What are the constraints imposed by the contexts?	How do the constraints cause design problems?	How do the constraints limit the solution options?
Trade-off analysis	What contexts can be compromised?	Can a problem be framed differently?	What are the solution options? Can a solution option be compromised?

Lamoreux, 2005) have been advocated.

Razavian et al. show that having an observer present during a design session to reflect can help improve design discourse quality (Razavian et al., 2016). Reflection requires a higher and deliberate level of thinking (Van Manen, 1977), but this is difficult as people tend to think naturalistically (Zannier et al., 2007; Moe et al., 2012).

The current issue for facilitating design reflection and reasoning is to find a suitable method that reminds designers to carry out that deliberate mode of thinking and reasoning. Reymen proposed to add reflective moments at the beginning and end of a session to discuss both the design situation and performed design activities (Reymen, 2001). To apply reflection in a software design practice and to remind designers to reason, Razavian et al. used a list of reflective questions aimed at improving design reasoning (see Table 1) (Razavian et al., 2016). These reflective questions were meant to be generic. Razavian et al. performed a case study in which they involved groups of students in 1 h practical sessions about a software design project. The test groups were asked reflective questions (see Table 1) by the lecturer, whereas for the control groups the lecturer was only passively present to answer technical questions the students might have had. It was concluded that active, externally induced reflection improves the quality of design reasoning.

The questions in Table 1 list the different reasoning techniques and associated reflective questions. The study by Razavian et al. has yielded encouraging results that reflective questions have a positive impact on the quality of design discourse (Razavian et al., 2016). However, some of the participants found the questions difficult to use as there are many of them. In order to remind designers to reflect on certain aspects of design without a complex list of questions, such as those listed in Table 1, we propose a simpler representation, a card system, to test its reflective power on designers and its impacts on design discourse.

3. Research method

The aim of our research is to find out if, and how, the use of reasoning reminder cards, based on common reasoning techniques, can influence and support design reasoning. This leads to our main research questions:

Q1) Does a reminder card approach influence designers and their reasoning in software architecture?
Q2) In what ways does the reminder card approach impact on design reasoning?

To answer these questions, we designed a set of reasoning reminder cards following the design science principles (Von Alan et al., 2004). The reasoning reminder cards simplify the reflective questions proposed by Razavian et al. (2016) into a set of cards. Instead of asking specific questions like 'Have you considered encrypting this communication channel?', we use seven symbols representing seven types of design rationale that we seek. By playing the reasoning reminder cards, the idea is that designers prompt each other: 'Have we considered such reasoning?'. The main reason why we simplify the questions in Table 1 into reasoning reminder cards, is the potential reduction of cognitive overload. When the participants are trying to familiarize with a complicated method (Sweller, 1988) they may not be able to focus on the design problems at the same time when applying the reflection methods.

Next, we set up a treatment experiment (Wohlin et al., 2012) to find if the cards prompt designers to reason more and find more design rationale using the reminder cards. In the experiment, teams of software designers need to collaborate to design a traffic light simulator (Petre and Van Der Hoek, 2013). Similar to the original setting, we assume that our subjects are capable of reasoning with this design problem despite their difference of experience and domain expertise. This assumption is made based on the nature of the problem situation. Traffic light and cars are familiar to most people but designing a technical solution to simulate this is a novelty to most designers (Petre and Van Der Hoek, 2013)

To investigate the effect of the reminder card approach, the participants are divided into two groups: a test group that uses the cards as treatment, and a control group without any treatment (Wohlin et al., 2012). The teams in the control group serve as baseline for comparison. To measure the effect of the card game, we recorded, coded and analysed each design session on reasoning and rationale techniques applied by the participants.

3.1. Treatment design: reasoning reminder cards to prompt reflective reasoning

We develop and test a simpler design reasoning method that prompts designers to reflect with less cognitive loading. Instead of having a lecturer or other external expert ask the reflective questions (e.g. as in (Razavian et al., 2016)), we use simple reasoning cards based on the reflective questions from Table 1, and include explicit "reflective moments" in the design discussion during which designers are encouraged to use these cards and reflect on their reasoning (Reymen, 2001). The first reason for this has to do with practical applicability of the method: we want a simple tool that software architects can easily be familiar with, and which does not depend on having specific experts in a design session as in the Razavian et al. experiments, or using special software, as is the case for many of the issue-based design rationale methods (Conklin et al., 2001). Reasoning reminder cards are a logical choice as it may stimulate creativity (IDEO 2015), stipulate learning about an architecture (Decisions, 2016), or perform planning (Grenning, 2012). The second reason for choosing a reminder card approach is to avoid the influence of an experienced person such as a lecturer whose knowledge and timing of asking relevant reflective questions may steer the student design groups to perform better. Finally, with the combination of (i) reminder cards that capture familiar design elements and; (ii) structured design sessions with reflective moments we intend to capture both aspects of Reymen's design method (Reymen, 2001), namely (i) the systematic analysis of design and (ii) reflective design sessions.

To design the cards, we follow the principles for design science (Von Alan et al., 2004). In a first iteration, we design the cards based on the available literature on design rationale in software architecture. As a next step, we validate the cards with a pilot group. Based on these outcomes, we design the final reasoning reminder cards which are used as treatment in the experiment.

In the first prototype, we designed 15 reasoning cards, three main cards representing the design activities (context, problem and solution), and one for each combination of design activity and design technique (context constraint, context assumption, problem constraint, problem assumption, and so on). With the prototype cards, we performed a pilot test. In this test, we gave two MSc students the 15 cards and the associated questions from Table 1, and asked them to design a simple traffic flow simulator based on the UCI experiment (Petre and Van Der Hoek, 2013). The students were told that they could play the cards at any time to start a relevant discussion on aspects of the design.

The main finding of the pilot session is that the cards were hardly played, and that their use tapered off over time. The participants mentioned that it was unclear exactly when to play the cards. Furthermore, it turned out that 15 cards were too many, and that the 12 combinations of design activities and techniques were too specific and hindered the design discourse – at one point the students themselves started discussing whether a particular constraint was a problem constraint or a solution constraint. The cards did have an influence on the design discussion as the designers immediately started using the terms used on the cards, such as assumption and constraint, to refer to points in their discussion. Summarizing, the students mentioned that they found the cards to be nice as a checklist, but distracting in use.

In order to simplify, we redesigned the approach into a deck of 7 cards, where each design activity and technique was represented by a separate card (Fig. 2). Furthermore, reflective moments were added into the design session to ensure that the cards were played, without restricting the full discussion to only the cards.

3.2. Experiment design

Given our main research questions, the objective is to find if the cards indeed prompt designers to reason more and find more design rationale, that is, whether the reminder card approach has a positive impact on the design reasoning process. This idea is reflected in our hypothesis:

The cards and the reflective moments used in a design session could improve the number of reasoning techniques used, and the number of design rationales found, by the designers.

We conducted an experiment to evaluate this hypothesis. The experiment was run with two different populations of designers, i.e. students and professionals. Participants need to collaborate in teams to design a traffic flow simulation program (Petre and Van Der Hoek, 2013) in a two-hour design session. Each population is divided into two groups: a test group who used the reasoning reminder cards during the design session as treatment, and a control group who designed without treatment. Each design discussion was recorded, transcribed and coded, and the use of the cards by the teams in the test group was logged.

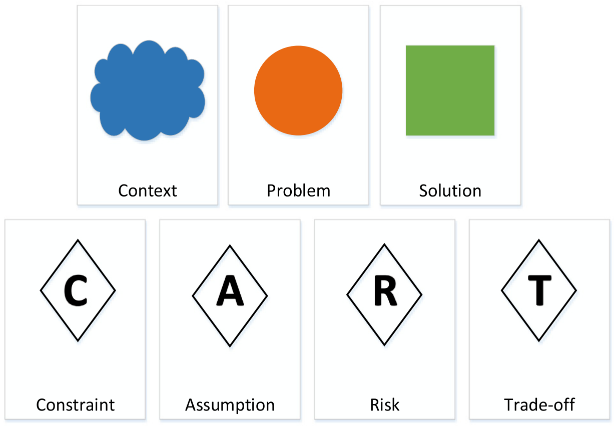


Fig. 2. The card deck of the reminder card approach.

After the design session, we asked the individuals of the test group to fill in a questionnaire, in which the participants were asked to evaluate the perceived usefulness on the use of the reasoning reminder cards, with questions like “Were the cards helpful during your discussion?”, and “Did the cards help you think of design issues and solutions you otherwise wouldn't have come up with?”. A detailed description of the experimental materials can be found in (Schriek, 2016).

3.2.1. Participants: students and professionals

For the experiment, we worked with two different populations of designers: students in a software architecture course at Utrecht university in the Netherlands and professional software designers in the United States and Australia. The student experiment involved 12 teams of students, with most having three designers; two teams had two designers, and one team had four designers. To divide the teams over the test and control group, we applied stratified sampling (Wohlin et al., 2012), to ensure both groups have an equal level of experience. For stratifying, the results of a previous software architecture assignment were used. As all students followed the same classes on modelling in software architecture, we controlled in this way the factors experience and familiarity with the subject as much as possible. Both the test and control group consists of 6 teams.

The experiment with the professional population involved 5 teams of 2 professionals each. The teams forming the control group were selected from the original Irvine experiment of which the transcripts were available (Petre and Van Der Hoek, 2013). These 3 teams consisted of 2 professionals each: Adobe (13.5 years average experience), Amberpoint (23 years average experience) and Intuit (13 years average experience). Details of the teams can be found in (Petre and Van Der Hoek, 2013). The design sessions of these three teams were conducted in the USA. The design task and the length of the design sessions are the same as the student control groups. For the test group, we recruited 2 professional test teams of 2 designers each in Australia from two software companies. The experiments were conducted in Melbourne Australia. One of the companies serves the financial sector (7 years average experience) and the other company serves the government sector (10 years average experience).

We justify using the control groups from the Irvine experiment because the experimental setup of our control groups is exactly the same as for the Irvine experiment: design a traffic simulation program in 2 h without any further special rules or guidelines. As such, we are able to analyse the transcript to work out what design rationale and how much design rationale they found without prompting by the card system. The results from the control groups can then be used to compare with the test groups in which participants were asked to use the card system.

3.2.2. Assignment and experimental procedure

The designers in our experiments were tasked with designing a traffic flow simulation program for a professor who teaches civil engineering (see (Petre and Van Der Hoek, 2013)). The professor wants to use the simulation to teach their students about traffic signal timing. Students must be able to create a visual map of an area by laying out roads, describe the behaviour of the traffic lights at each of the intersections, and change and simulate traffic flows on the map. The designers have to focus on the interaction that the students will have with the system, and to describe the system's runtime functional elements.

The designers worked in teams for 2 h on the assignment, during which they had to discuss and design the traffic simulator. The teams in the control group were given no specific instructions on exactly how

Table 2
Coding scheme for the design discussion transcripts.

Category	Code	Description
Design reasoning techniques		
Problem structuring	PROB	Identifying and discussing the key issues of the design
Trade-off analysis	TA	Weighing the pros and cons concerning the design to come to a decision
Option generation	OG	Discussing the options available for design solutions
Assumption analysis	AA	Questioning the premises of the requirements and context, the validity of arguments
Constraint analysis	CA	Identifying constraints in the design and how these constraints influence the design
Risk analysis	RA	Identifying risks in the design and how to mitigate those risks
Design reasoning outcomes		
Design rationale		
Pro	PRO	Argument for a proposition
Con	CON	Argument against a proposition
Constraint	C	A restriction on the condition of the design
Assumption	A	A supposition that is taken for granted or questioned to determine the validity
Risk	R	An aspect of the design which is identified to be a threat to achieving the system goals
Other design reasoning outcomes		
Option	DO	A solution option
Issue	DI	A design problem
Decision	DD	Evaluating solution options to make a decision

they should behave during the design session. The participants in the test group each received a desk with the seven rationale reminder cards (Fig. 2) as treatment, and were told to use the cards during the design discussions. The students in the test group were also presented with the table of reflective questions (Table 1) as a reference. There were no real rules for playing the reminder cards, but it was mentioned that the card could be played during discussion – for example, when you mention a risk you can play the risk card. The control groups were given the same exercise without any additional instructions. The design discussions were recorded and transcribed, and the use of cards by the teams in the test group was logged.

To validate the setup of the experiment, we performed a pilot with a team of three students. The pilot showed that the students in the heat of discussion forgot to play the rationale reminder cards. To stimulate the use of the cards, we decided to add three reflective moments when the teams were specifically asked to play the cards, following the design method proposed by Reymen (2001). The three reflective moments are spread out across the two-hour design session, at 15 min, 45 min and 1 h and 45 min into the session.

3.3. Analysis of the design sessions

Of each design session the audio was recorded and afterwards transcribed by two students. These recordings and transcripts form the basis of our analysis. Additionally, each team in the test group logged when a card was placed or removed from the session, and by whom. The raw recordings and transcripts would give us *design session length, cards played, design reasoning terms used* and *design rationale created*. The design session length is based on the length of the recordings, and the number of cards played follow from the logs of the student test teams. One of the authors logged the cards played by the professional teams.

A first method to analyse the *design reasoning terms* used by the different teams, is to count the frequencies of the exact terms from the cards in the transcripts, such as “context”, “constraint”, “assumption” and so on (see Fig. 2). Additionally, we looked for “assume” and “rule” as synonyms for “assumption” and “constraint”, respectively. The term frequency could simply be counted in the transcripts. Counting the design reasoning terms used does not tell the whole story, as a group may be talking about a constraint without explicitly mentioning the word “constraint”. For example, if a designer states “The simulator should be able to simulate traffic flows on a map”, the designers are performing constraint analysis (i.e. a *design reasoning technique*), without mentioning the technique explicitly. One result of applying design reasoning techniques is to generate design outcomes. This outcome can be a specific constraint, which is an example of design rationale, or other design outcomes such as design solution options. When encoding the transcripts, we were cognisant that the speaker may or may not mention the exact word or a synonym. It is for this reason that we encoded the transcripts as *design reasoning technique* or *design rationale outcome*, based on the meaning of the speaker.

Although we asked the test teams to use the *context* card, we do not analyse this aspect in this research, as the concept of contexts is complex and not so obvious to the participants. In the coding, we interpret context as the basic information that drives a decision. Examples include requirements and any environmental factors. Preliminary work on defining context as a viewpoint has recently been published in the software architecture field (Bedjeti et al., 2017). In our experiment, we focus mainly on the analysis of design reasoning and design rationale.

The transcripts of each team's design discussion were coded by two separate coders using the NviVo 10 qualitative data analysis software (Bazeley and Jackson, 2013). For coding the transcript, a coding scheme was used in which both *design reasoning techniques* and *design rationale* were included (see Table 2). We imposed several coding rules to ensure coding consistency between the coders: (i) a design decision follows a design issue; (ii) a design issue is not a requirement, but a stated problem that needs to be solved; (iii) option generation needs at least two options; and (iv) a trade-off analysis needs at least one pro and con.

Given the coded transcripts we analyse and compare the number of design reasoning techniques used and design rationales mentioned by the test and control groups. This allows us to test the following null hypotheses:

Ha_0: The number of design reasoning techniques applied is not higher for the test groups, than for the control groups.

Ha_0 can be tested for each of the individual techniques – assumption analysis, constraint analysis, risk analysis, problem structuring, trade-off analysis and option generation – as well as for the total number of techniques used.

Hb_0: The number of distinct design rationale mentioned is not higher for the test groups, than for the control groups.

Hb_0 can be tested for each of the individual types of rationales – assumption, constraint, risk, issue, decision and trade-off arguments (i.e. pro and con)– as well as for the total number of rationales.

In the test group, the reminder card approach does not involve specific rules on how the cards should be used in the discussions or during the reflective sessions; the participants were simply told they could discuss their design “using the reminder cards”. Thus, in the event that hypotheses Ha_0 or Hb_0 can be rejected, the question is whether the increase in design reasoning techniques used and/or rationales mentioned is due to the fact that test groups *actually played the cards.* It may be that even though they did not play any cards, they still used more techniques. It may be that the mere presence of the reminder cards in front of them encouraged participants to reason more. We test this correlation in Hc_0.

Hc_0: The number of cards played during design sessions is not positively correlated with the number of design reasoning techniques applied.

The practice of reasoning techniques may or may not generate the desired outcomes that help the course of design. We postulate that the more reasoning techniques the designers apply, the more design rationale can be generated. Repeated mentioning of the same design rationale is not very useful, so we are interested in the distinct design rationale outcomes. We test the following hypothesis to evaluate whether design reasoning is positively correlated with the generation of distinct design rationale.

Hd_0: The number of reasoning techniques applied is not positively correlated with the distinct design rationales.

In addition to hypotheses testing, we also use descriptive statistics and qualitative analysis to analyse how the teams in both groups reason with their designs.

4. Research result

The design session of each team has been transcribed. Two researchers coded the resulting transcriptions. To ensure that the interpretation of the design reasoning techniques and design rationales was consistent among the coders the inter-coder agreement was checked by calculating Cohen's Kappa (Cohen, 1968), which varied between transcripts (lowest 0.6, highest 0.76). The average Kappa was 0.64, indicating substantial agreement between the two coders.

4.1. Design session length and word count

All teams were each given two hours to complete their design session. For each of the teams, we measured the time taken for the design sessions (Appendix A). Of the teams in the control group, four groups spent around 2 h (m = 1:36 h, sd = 0:23 h), while three teams spent between 1 and 1.5 h and one team spent just below one hour. The teams in the test group spent on average almost the full time available on the assignment (m = 1:55 h, sd = 0:17 h). Only one team finished far before the two-hour limit (1:23 h), but from the transcripts it became clear that this was due to miscommunication in the team about how much time they had left. The difference between the times of the teams in the test and control group is not significant ($p = 0.0833$, Mann–Whitney test).

The teams in the test group were told to have a reflective moment at 0:15 h, 0:45 h and 1:45 h. The two professional test teams PT1 and PT2 and student test teams ST1, ST5 and ST6 used the three moments. Student teams ST2 and ST3 used only two reflective moments (even though ST3 spent two hours on the assignment), and ST4, who spent 2:24 h on the assignment, had four reflective moments. In addition to measuring the time we also measured the word counts of the student team transcripts as an estimate of how much has actually been said during the design sessions (Appendix A). Teams in the test group ($m = 13{,}299$, sd = 1762 words) talked more than teams in the control group ($m = 8581$, sd = 2382 words), and this difference is significant ($p = 0.0021$, Mann-Whitney test). We further discuss these results in Section 5.4.

4.2. Card play frequencies

Focusing on the teams in the test group, and how these teams play the cards, we can see that *Assumption card* was generally the most popular card, followed by *Problem card. Solution card* was mainly used by student team ST4, who often paired the card with *Problem.* Each professional team played all cards at least once, whereas most student teams tended to ignore one or more cards (only ST5 and ST6 used all cards). Furthermore, the professional teams played more cards ($m_{prof} = 43.5$; $m_{stud} = 11.7$), but there was a big difference between the two professional teams: one team played 20 cards, while the other played 67 cards ($sd_{prof} = 33.2$). The students were generally more consistent, with totals number of cards played ranging from 6 to 16 ($sd_{stud} = 4.5$). Fig. 3 shows the descriptive statistics of cards played by all test teams.

4.3. Design reasoning terms used

The explicit design reasoning can be observed when the participants used particular terms that were introduced to the teams in the test group, through the introduction and the reminder cards. The teams in the control group were not exposed to these terms. We examine the use of these terms during design discourse. Fig. 4 shows that the teams in the test group use reasoning terms from the cards more frequently, whereas teams in the control group hardly use any of the terms. It shows that the cards appeared to invite the designers to directly use those terms. The only terms mentioned by the teams in the control group are *constraint* and *assumption. Constraint* was mostly used by the student teams in the control group (20 out of 28 mentions), and *assumption* was mostly used by the professional teams in the control group (27 out of the 30 mentions).

4.4. Design reasoning techniques

The use of reasoning terms is a reflection of using the design reasoning techniques. However, when a design reasoning dialogue takes place, many terms can be used. For instance, while discussing the solution for traffic simulation, team SC6 (Fig. 5) addresses several options and assumptions, without mentioning these terms explicitly. This shows that in reasoning, sometimes several reasoning terms can be used in reasoning with one issue, and sometimes reasoning can take place without using an exact term. To determine which reasoning techniques took place, we encoded the design dialogues of all the teams (see Table 2).

After coding the transcripts with the codes from Table 2, we counted the design reasoning techniques from all the teams used during their session. The totals of all teams in the same group for each type of techniques are shown in Fig. 6.

In total, the six student teams in the test group used the techniques 340 times and the six student teams in the control group used the techniques 194 times. The two professional teams in the test group used the techniques 134 times in total, and the three professional teams in the control group used the techniques 128 times in total. Table 3 shows the means and standard deviations for the student and professional teams for each reasoning technique. We tested hypothesis Ha_0 for each of the difference of the use of reasoning technique between the teams in the test and control group. Analysing the data shows that the data is not normally distributed, and that there are outliers. We therefore decided to perform non-parametric Mann-Whitney U tests ($\alpha = 0.05$) to determine whether the use of reasoning techniques between the test and control group are statistically significant, as this test makes few assumptions about the distribution of the data, and reduces the effect of outliers and heterogeneity of variance. Furthermore, we only performed statistical testing for the student teams (6 test and 6 control) but not the professional teams as there were too few professionals teams (i.e. 2 test and 3 control groups), and they were unequally distributed.

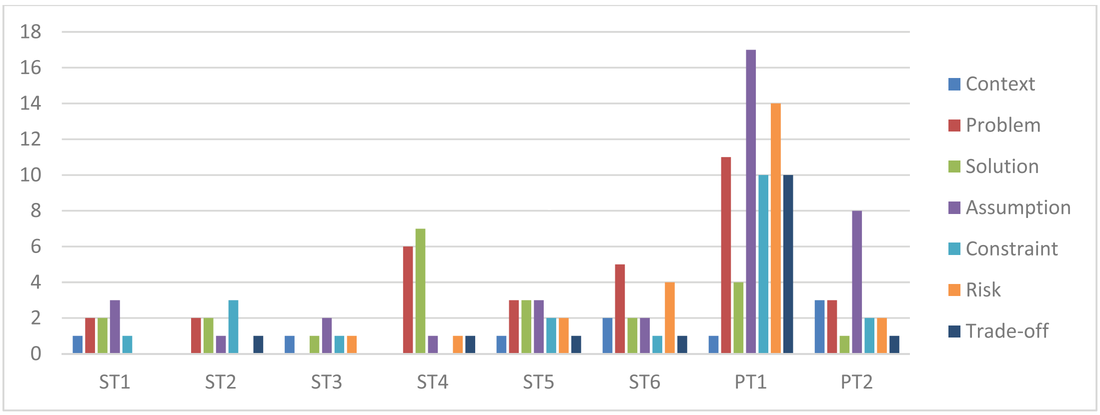


Fig. 3. Reminder cards played by test groups.

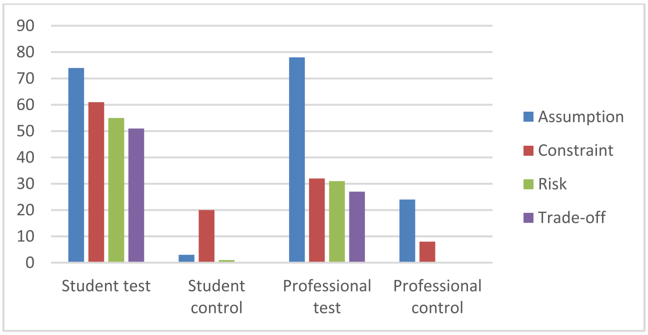


Fig. 4. Total design reasoning terms used by all teams.

The results of the tests indicate that there are noticeable differences between the student teams in the test and control group: on average, all reasoning techniques are performed more by the teams in the test group, who used the reminder cards, than by the teams in the control group. We accepted the null hypothesis for constraint analysis and option generation, meaning that there is no significant difference between the student teams in the test and control group. We rejected the null hypothesis for all the other reasoning techniques, meaning that there is a significant difference between the number of reasoning techniques used by teams in the test and control group (see column P_{m-w} in Table 3). Table 3 also shows the means and standard deviation of each technique used.

For the student teams there is a significant difference between the test and control group for the amount of assumption analysis, risk analysis, trade-off analysis and problem structuring undertaken. These reasoning techniques are used significantly more by teams in the test group than by teams in the control group. For constraint analysis and option generation there is no significant difference. Because on four of the six reasoning techniques the teams in the test group score significantly higher, we also find significant results when comparing the total number of techniques used by each team.

For the professional teams, we see that the average amount of reasoning techniques used is fairly high. The professional teams in the test group have the highest average of all the types of teams, with a fairly low standard deviation. The professional teams in the control group have a higher average for all the reasoning techniques when compared to the student teams in the control group. Whilst these results are not statistically tested and thus we cannot generalise, they appear promising when considering the application of the reminder cards in a real design setting.

4.5. Design reasoning outcomes

We analysed the transcripts to reveal that design discussions are typically overarching. It means that the outcome(s) such as those classified in Table 2 may take place in multiple discussions, and one discussion may cover multiple outcomes. Thus, even though one team may have carried out many design reasoning techniques, they may not generate more design reasoning outcomes or design rationale if they

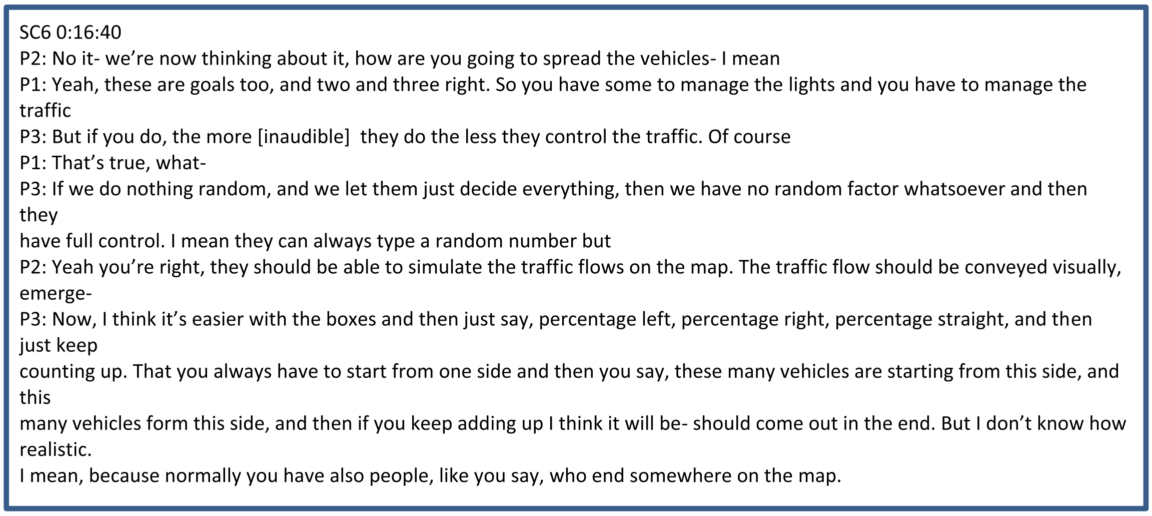

SC6 0:16:40
P2: No it- we're now thinking about it, how are you going to spread the vehicles- I mean
P1: Yeah, these are goals too, and two and three right. So you have some to manage the lights and you have to manage the traffic
P3: But if you do, the more [inaudible] they do the less they control the traffic. Of course
P1: That's true, what-
P3: If we do nothing random, and we let them just decide everything, then we have no random factor whatsoever and then they
have full control. I mean they can always type a random number but
P2: Yeah you're right, they should be able to simulate the traffic flows on the map. The traffic flow should be conveyed visually, emerge-
P3: Now, I think it's easier with the boxes and then just say, percentage left, percentage right, percentage straight, and then just keep
counting up. That you always have to start from one side and then you say, these many vehicles are starting from this side, and this
many vehicles form this side, and then if you keep adding up I think it will be- should come out in the end. But I don't know how realistic.
I mean, because normally you have also people, like you say, who end somewhere on the map.

Fig. 5. Extract of transcript SC6.

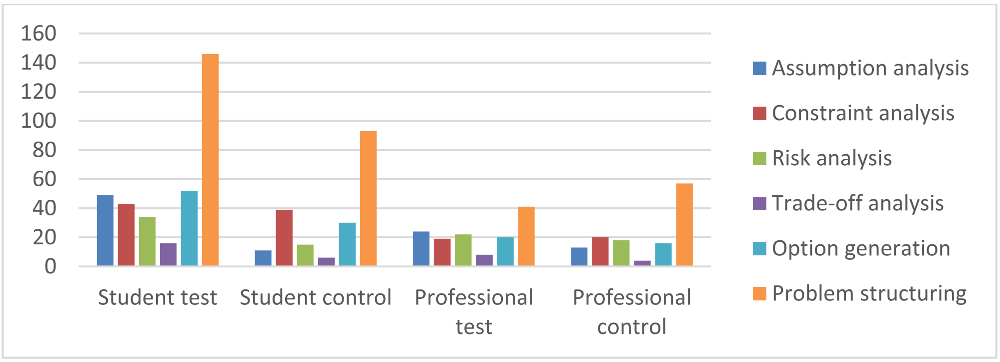


Fig. 6. Total design reasoning techniques used by all groups.

keep repeating the same thing. To study whether more design rationale was created through reasoning, we only counted the *distinct* pieces of design rationale, so if, for example, a team discusses the same constraint three times at different points during the two-hour assignment, this was only counted once. The total numbers of distinct design rationale are shown in Fig. 7.

Table 4 shows the means (m) and standard deviations (sd) of the distinct design outcomes, for both the test and control group of the student and professional teams. We performed non-parametric Mann–Whitney U tests ($\alpha = 0.05$) to see if the student teams in the test group are statistically different from the student teams in the control group. The *p*-values in Table 4 shows that the teams in the test group yielded significantly more design outcomes than the teams in the control group, except for constraints, pros, issues and options. We therefore reject the null hypotheses Hb_0 for design rationale assumption, risks and con. This result mirrors the differences in the application of *assumption analysis* and *risk analysis* by the student teams in the test and control groups (cf. Table 3). Similarly, the number of distinct *constraints* mentioned was not significantly higher for the student teams in the test group, which matches the earlier finding that there is no significant difference in the use of *constraint analysis* for the student teams in the test and control group.

Trade-off analysis was used significantly more by the student teams in the test group. Trade-off analysis requires evaluating multiple *pro* and *con* arguments. When looking at the total number of arguments (*pro* and *con*), we see that the student teams in the test group put forth significantly more arguments than the teams in the control group. However, there is a difference in the use of *pros* and *cons* between the student teams in the test and control groups. For the *cons*, the difference is statistically significant but not for the *pros*.

Looking at *design issues, options*, and *decisions* (Table 4), we see that while these design outcomes are on average higher for the student teams in the test group, the differences are not significant. Finally, when we examine the total number of design outcomes mentioned, we see that the student teams in the test group outperform the teams of the control group, but the difference is not statistically significant ($p = 0.055$).

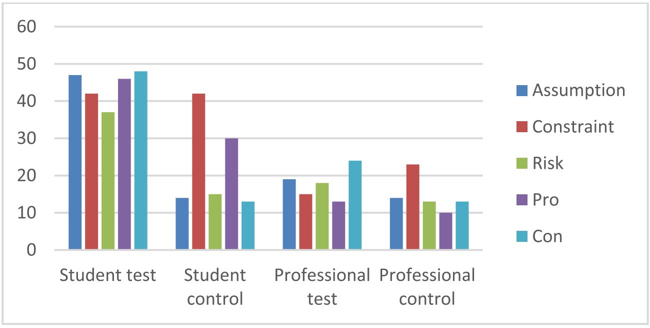


Fig. 7. Total number of distinct design rationale identified by all teams.

4.6. Correlation between cards played, the use of reasoning techniques and design rationale

We test Hc_0 and Hd_0 to understand if the cards played are correlated with the number of design reasoning techniques applied; and whether the number of design reasoning techniques applied is correlated with the number of distinct design rationale generated. Fig. 8 shows a scatter plot depicting the cards played by the teams and the reasoning techniques applied in their design discourse (notice that some data points overlap). The scatter plot indicates that the design reasoning techniques and cards played are not correlated. Similarly, Fig. 9 shows a scatter plot of the reasoning techniques applied and the design rationale.

We use Spearman's rank correlation for the statistical analysis due to the nature of the data, which is not normally distributed and contains outliers (see Table 5). We accept the null hypothesis Hc_0 for the overall cards played correlated with reasoning techniques. There is no correlation between the total cards played per team and the total number of design reasoning techniques ($r = 0.594$, $p = 0.121$). We accept all of the null hypotheses for the cards played and reasoning techniques used except for assumption. We interpret this result in Section 5.

There are correlations between the number of design reasoning techniques applied and the number of distinct design rationale for some

Table 3
Comparing design reasoning techniques used by teams in the test and control groups.

	Student groups					Professional groups			
	m_{test}	sd_{test}	$m_{control}$	$sd_{control}$	$p_{M\text{-}W}$	m_{test}	sd_{test}	$m_{control}$	$sd_{control}$
Assumption Analysis	8.17	3.19	1.83	1.17	0.004	12.00	2.83	4.33	2.08
Constraint Analysis	7.17	2.79	6.50	2.07	0.380	9.50	3.54	6.67	2.08
Risk Analysis	5.67	0.82	2.50	1.05	0.004	11.00	7.07	6.00	5.20
Trade-off Analysis	2.67	1.51	1.00	1.10	0.045	4.00	2.83	1.33	1.53
Option Generation	8.67	5.85	5.00	2.90	0.260	10.00	5.66	5.33	1.15
Problem structuring	24.33	4.97	15.50	3.02	0.008	20.50	2.12	19.00	1.00
Total	56.67	13.94	33.50	4.89	0.004	67.00	3.50	42.67	8.33

Table 4
Comparing distinct design outcomes generated during design discourse.

	Student groups					Professional groups			
	m_{test}	sd_{test}	$m_{control}$	$sd_{control}$	$p_{M\text{-}W}$	m_{test}	sd_{test}	$m_{control}$	$sd_{control}$
Assumptions	7.8	3.6	2.3	1.5	0.005	9.5	0.7	4.7	2.3
Constraints	7.0	1.5	7.0	2.0	0.631	7.5	2.1	7.7	2.1
Risks	6.2	3.7	2.5	1.0	0.004	9.0	5.7	4.3	3.2
Arguments	18.3	9.2	8.2	5.6	0.025	30.0	17.3	9.0	5.3
Pro	7.7	5.3	5.0	3.6	0.378	8.7	4.5	3.3	3.5
Con	8.0	3.7	2.2	1.6	0.016	16.0	9.8	4.3	3.1
Issues	15.7	7.5	11.7	5.7	0.423	22.7	10.6	12.3	3.1
Options	24.7	12.0	15.8	7.7	0.200	40.7	21.6	19.7	7.6
Decisions	15.3	7.7	12.3	5.7	0.689	18.0	7.9	11.7	3.5
Total	84.8	26.5	50.5	20.8	0.055	109.5	50.2	69.3	18.7

reasoning techniques (see Table 5). For the teams in the test group, both for the student and professional teams, we rejected the null hypotheses Hd_0 for assumption and trade-offs. The test for risk is marginally rejected. It shows that when the teams in the test group discussed more about the assumption, risk and trade-offs, they found more distinct design rationale. When we analysed the correlation between the use of design reasoning technique and design rationale for both the test and control group, we found that the correlation between them are strong for all types of design rationale except constraint. We interpret this result in Section 5.

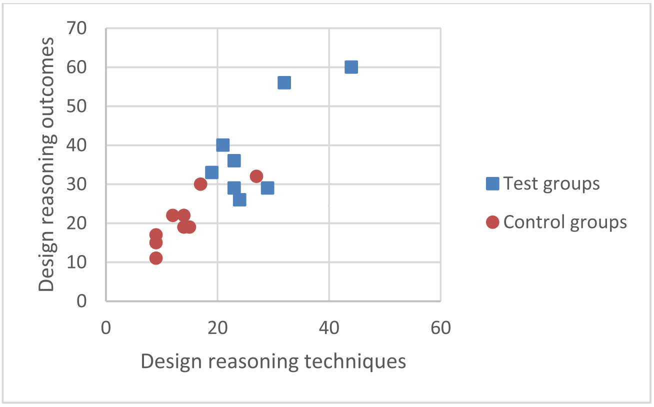


Fig. 9. Scatter plot of design reasoning techniques and design rationale.

4.7. Participant questionnaires

We provided a questionnaire to all the participants in the test group (17 students and 4 professionals). The first question was “Did the questions table help you understand the cards meaning, or did you not use it at all?”. Of the participants, 7 students and 2 professionals answered with “yes”, 7 students said they only used in sometimes or in the beginning, and 3 students and 2 professionals said they did not use the table at all.

The second question was “Were the cards helpful during your discussion?” In total, 10 students and all 4 professionals indicated they found the cards helpful for different reasons (e.g. “thought more deeply about problems”, “helped lower tension in discussion”, “helped structure the discussion”). 3 students indicated they did not find the cards helpful. Of the participants that found the cards helpful, 4 students indicated they only found them helpful in the beginning or only for some cases.

The third question was whether the cards felt like they were getting in the way. 8 students and 1 professional felt this was the case, saying things such as “When we were forced to use them the discussion stalled a little”, “[cards] got in the way and slowed down process” and “[cards] became restrictive later”.

The fourth question was “Did the cards help you think of design issues and solutions you otherwise wouldn't have come up with?”. 5 students and 1 professional answered with an unequivocal “yes”, 7 students and 2 professionals indicated that the cards helped them come up with issues and solutions, but only in the beginning or for some types of rationales (e.g. risks or assumptions), and 5 students and 1 professional indicated the cards did not help them to come up with any new issues or solutions.

The fifth question was “Do you have any suggestions to improve the Design Support Card game?”. Only 6 students and 1 professional answered this question, 4 students and the 1 professional indicating that it would be good to have more rules for the game (e.g. on when to use which cards), 1 student indicating that he would have liked more time, and 1 student would have liked the cards to be integrated into the discussion more naturally.

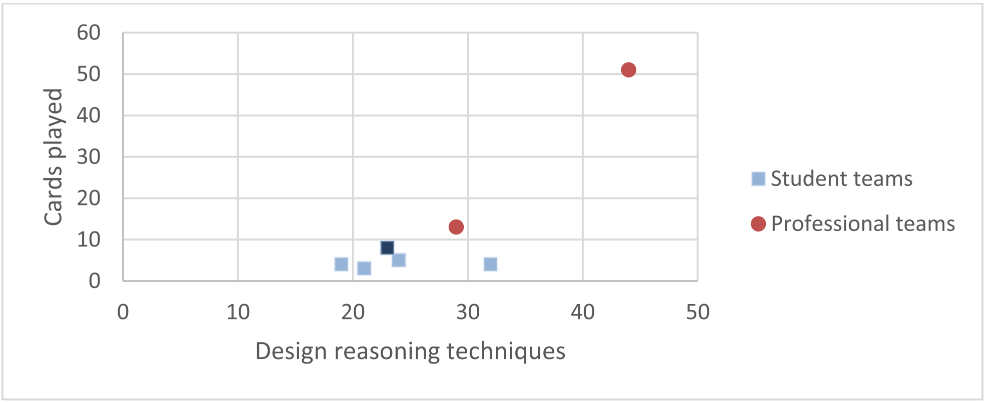


Fig. 8. Scatter plot of cards played and design reasoning techniques (darker shade indicates more than one team with same results).

Table 5
Hypotheses testing results: cards played and design reasoning techniques used, design reasoning techniques used and design rationale found.

	Correlations between cards played and reasoning techniques used (all test groups)	Correlations between reasoning techniques used and distinct design rationale found (all test groups)	Correlations between reasoning techniques used and distinct design rationale found (both test and control groups)
Assumption	0.835 (p = 0.010)	0.855 (p = 0.030)	0.987 (p = 0.000)
Constraint	0.604 (p = 0.113)	0.704 (p = 0.118)	0.644 (p = 0.084)
Risk	−0.076 (p = 0.857)	−0.026 (p = 0.961)	0.932 (p = 0.007)
Trade-offs	0.154 (p = 0.716)	0.849 (p = 0.033)	0.501 (p = 0.311)
Overall	0.594 (p = 0.121)	0.319 (p = 0.537)	0.858 (p = 0.029)

5. Analysis and discussion

The main lesson learned from this study is that the reminder card approach influences design reasoning and design discourse. In this section, we analyse the results presented in Section 4.

5.1. How reminder cards influences design discourse

5.1.1. Use of reasoning techniques by test and control groups

We used statistical analysis to check if the cards made any difference to the use of design techniques during design discourse. The average reasoning techniques used by the teams in the test group are higher than the teams in the control group. Student teams in the test group used the techniques, on average, 56.7 times versus the teams in the control group 33.5 times. The professional teams in the test group used the reasoning techniques on average 67 times versus 42.7 times by the teams in the control group. There is a difference in the way the two groups discussed design. The reminder cards have changed the way design discourse was conducted in terms of the use of reasoning language. To a different extent, this kind of changes of how designers work have been observed in other research works as well (Tang et al., 2008; van Heesch et al., 2013). Works such as (Lytra et al., 2015; Van Heesch et al., 2012) used architecture knowledge, viewpoints, etc to induce design reasoning and better decision making, and they also made similar observations that methods can help designers to reason better.

Statistical testing on the difference of use of reasoning techniques between the teams in the test and control groups (Ha) shows that overall there is a significant difference in the general use of reasoning techniques, i.e. when all techniques are counted. This indicates that during design discourse, student teams in the test group are more cognisant of design reasoning. We tested the significant difference for each of the reasoning techniques individually. Most reasoning techniques are significantly different, except for constraint and option generation. From the transcript, we have hints that the student teams in the test group discussed constraints based on requirements. So the identification of constraints was mainly based on requirements. As such, their reasoning in this regards was not statistically different from the teams in the control group. The student teams, both in the test and control group, were also not very effective in generating solution options (see Table 4), even though the student teams in the test group talked more about design options (see Table 3).

5.1.2. Impact of reminder cards on design reasoning techniques

We tested hypothesis Hc_0 to see if there is any correlation between the number of times cards are played, and the reasoning techniques that are used. As shown in Table 5, except for assumptions, there are hardly any correlations between cards played and reasoning techniques used. This does not mean that the reminder cards have no effect. From the transcript, it appears that the participants use the cards mentally instead of physically. During the experiments we observed that the concepts of the cards, the techniques they refer to, are increasingly incorporated into the discussion even when the reminder cards were not played physically. The participants simply looked at the cards to serve as a visual reminder. We did not count that as card played. This is a plausible argument explaining why no correlation could be found quantifying the significance of the cards played. The rejection of hypothesis Ha_0 together with accepting hypothesis Hc_0 clearly indicates that the reminder cards have an impact, except that the participants do not necessarily play the cards physically.

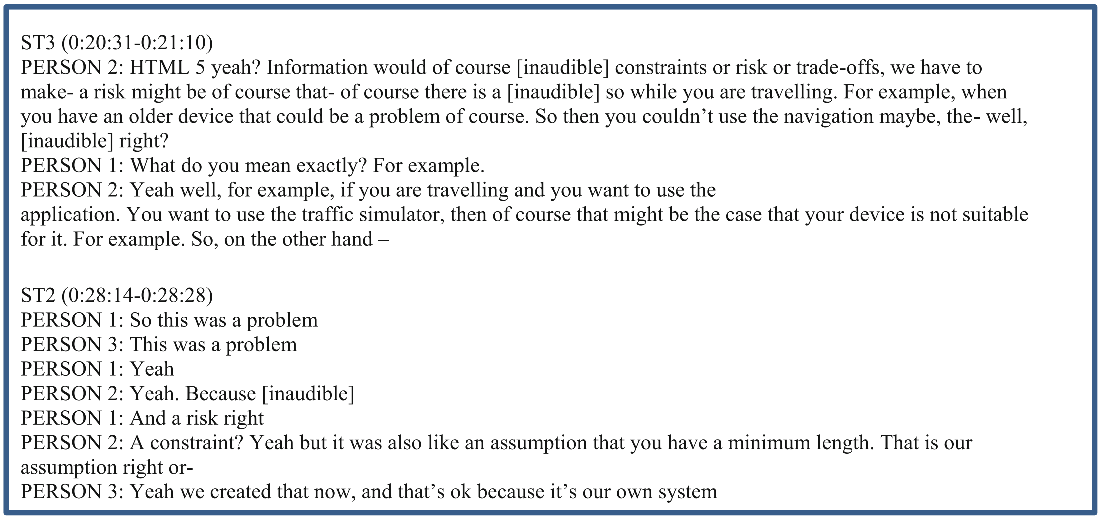
5.1.3. Design dialogue and questionnaire feedback

From the analysis of the transcripts, we learned that the reminder cards directly influence the design discourse in two ways. Firstly, the reminder cards provide inspirations for participants to investigate a certain reasoning topic. Secondly, the students use the reminder cards to reassess their previous discussion by classifying it in card terms, e.g. a system rule is later identified as having been a constraint. Examples like the extract from ST3 show how these cards are used for inspiration (Fig. 10). Person 2 was looking over the cards searching for issues to discuss and came up with a risk, which needed to be clarified for the other person. This risk made the designers reconsider an earlier assumption, that the program is a web-based application, which later turned into a nearly 5 min long trade-off discussion. Excerpts from ST2 show us that the cards were being used for classification (Fig. 10). Here they discussed a problem and found a solution for it. But when they reassessed the discussion as a problem, they realized that in order to solve the problem, there were other risks and assumptions as well.

Additionally, we asked the participants in the test group to fill in a questionnaire after the design exercise to study how they perceived the reminder cards (Section 4.7). The professionals of both teams in the test group are mostly positive about the use of the reminder cards and perceived them as being helpful during discussions to remind the designers what to look out for, as they all indicated that they found the cards helpful as reminders or goto points in the discussion. Opinions among the professionals in the test group were mixed on whether the cards helped to find more design issues. Two professionals (one from PT1 and one from PT2) stated that the cards prompted them to look for issues which might otherwise not have been looked at, but the other two stated that the issues found were things they always looked at, even without the reminder cards.

Of the students, the majority perceived the cards as being helpful, especially in the beginning, for structuring the discussion and thinking more deeply about the problems (10 out of 17 students), whereas only 3 students said they did not find the cards helpful. About half of the students also felt that the current structure with the reflection moments in which they were forced to use the cards interrupted the natural flow of discussion. On the other hand, four students wanted more structure and rules in the use of the reminder cards, such as a mandatory rule to use all the cards at least once. Finally, the majority of students (12 out of 17) indicated the cards helped them find issues they would not otherwise have found, at least for some types of issues.

The positive and more negative students are mostly evenly distributed among the five different student teams. Interestingly, the best performing student team in terms of techniques used and outcomes found (ST1 with 84 techniques used and 156 distinct outcomes found) were least positive as a team – not one of the three team members indicated that they really found the cards helpful, and they only played 8 cards in total. However, the members of ST1 were very clear in that


ST3 (0:20:31-0:21:10)
PERSON 2: HTML 5 yeah? Information would of course [inaudible] constraints or risk or trade-offs, we have to make- a risk might be of course that- of course there is a [inaudible] so while you are travelling. For example, when you have an older device that could be a problem of course. So then you couldn't use the navigation maybe, the- well, [inaudible] right?
PERSON 1: What do you mean exactly? For example.
PERSON 2: Yeah well, for example, if you are travelling and you want to use the application. You want to use the traffic simulator, then of course that might be the case that your device is not suitable for it. For example. So, on the other hand –

ST2 (0:28:14-0:28:28)
PERSON 1: So this was a problem
PERSON 3: This was a problem
PERSON 1: Yeah
PERSON 2: Yeah. Because [inaudible]
PERSON 1: And a risk right
PERSON 2: A constraint? Yeah but it was also like an assumption that you have a minimum length. That is our assumption right or-
PERSON 3: Yeah we created that now, and that's ok because it's our own system

Fig. 10. Reasoning discussions using the cards.

they used the provided table with questions (cf. Table 1) intensively during the discussion, indicating that while their discussion was not guided by the cards themselves, they did reflect on the concepts of the card game.

5.2. Design rationale

The main purpose of the reminder cards is to prompt designers to consider design reasoning and uncover more design rationale. If the reminder cards make any impact, we should be able to detect the results by measuring the amount of design rationale outcomes that are produced. If the reminder cards have a positive impact, then the teams in the test group should produce more design rationale. The results of the experiment have shown that the teams in the test group discussed risks, assumptions and trade-offs much more than teams in the control group, for both students and professionals. As a result, the teams in the test group found more distinctly identified risks, assumptions, and pros and cons. Pros and cons are qualifications of different solution options when making trade-offs (Table 4).

Except for constraints, professional teams in the test group identified many more design rationale than the teams in the control group. In the testing, we accept Hb_0 for constraints (i.e. insignificant difference), and reject Hb_0 for all other design rationale, i.e. the difference is statistically significant. Professional teams in the test group found more distinct risks, assumptions or pros/cons (Table 4). The professional teams in the test group also identified many more distinct risks, assumptions and pros/cons than all student teams and professional teams in the control group. Without the reminder cards the professional teams in the control group identified 72% more risks than the student teams in the control group. This may be attributed to experience. With the reminder cards, the professional teams in the test group identified 45% more risks than the student teams in the test group. This result appears to suggest that (a) the reminder cards alone help to generate more design rationale; and (b) the reminder cards and professional experience combined have a greater effect on the generation of design rationale.

Assumption is clearly not what the teams in the control group considered consciously. This is shown by their low number of identification as compared to the teams in the test group (Table 4). We observe that the reminder cards played a significant role in reminding the teams in the test group to check their assumptions, resulting in many more assumptions identified.

We tested hypothesis Hd_0 to see if there are correlations between design reasoning and design rationale. We tested this hypothesis for all teams in the test group as well as for all teams combined (see Table 6). With all teams in the test group (i.e. 6 student test groups and 2 professional groups), we found that only assumptions and trade-off analysis are correlated with their corresponding design rationale. However, when we test the correlation between the use of design reasoning techniques and design rationale for all teams (i.e. 8 teams in the test group and 9 teams in the control group), we see significant correlation in all design reasoning techniques. This means that the more reasoning techniques are used, the more distinct design rationale outcomes are generated.

Two reasoning techniques are not correlated with the design rationale outcomes, being constraints and risks. We observed that the professional teams in the test group discuss more on constraints than professional teams in the control group, but that discussions did not result in identifying more constraints. Despite the difference in the amount of constraint statements made by the professional teams in both the test and control groups, the distinct number of constraints identified by professional teams (both in the control and test groups) are about the same (Table 4). The amount of discussions for the student teams in the test and control groups is almost the same (Table 3). This raises the question why the reminder card approach does not improve constraint analysis. One possible explanation for this is the very nature of constraints, being defined as a limiting condition that a design concern imposes upon the outcome of a design decision (Holyoak and Simon, 1999). In our experiment, constraints are initially bound to the requirements. The subjects of the experiments have not had prior

Table 6
Improvements on design issues and solution options identified.

	Improvement between student test and control groups	Improvement between professional test and control groups
Average problem structuring technique used	56.7%	7.9%
Average solution option technique used	74%	88.7%
Average distinct problems identified	34%	84%
Average distinct solution options identified	563%	106.5%

experience with designing such a simulator. When thinking about this design and what this system must accomplish, designers think about what is required and what is not required. The constraints initially come from the given requirements. All teams from the test and control group identify constraints as things that are not allowed or rules that the system must follow stated by the design briefing. What is interesting here is that all teams identify many of the same constraints, of which many are directly taken from the design briefing, even using the same wording. We find that teams both in the test and control group frequently used the literal requirements presented in the text as constraints. The use of constraints for reasoning may be more noticeable if the design teams had the opportunity to delve more into the technical and implementation design.

The correlation between risk reasoning and identification of risk are significant for all teams (control and test combined) but insignificant for the teams in the test group only. This may be due to few data points available for the test group, as shown in the scatter plot depicted in Fig. 11 (notice that some data points overlap). It may also be because that there are differences in the capabilities of the teams in the test group in employing reflection and reasoning. An example is the differences of non-unique reasoning terms between PT1 and PT2 (see Appendix C). PT1 is more verbose with risk and trade-offs than PT2.

The extract of ST4 discussion shows part of a larger trade-off analysis they did. Several options were heavily discussed, mostly to have either a standalone program, or one which is cloud or web-based (Fig. 12). In this discussion, person 1 mentioned that a pro for a cloud based program would be that you can update every hour, but that a con is that a powerful but costly server is required. The person then proceeded to suggest another option, for the user to pay for the usage of the server. This was not well-received and person 1 admitted that this option would still be a very expensive one and gave a pro to their first option, to use a local standalone version to which the others agreed. Even though the team eventually went with their first option, they took the time to explore multiple options and critically assessed them by providing both pros and cons. Teams in the control group did not do this often.

The amount of reasoning and distinctive risks, assumptions and trade-offs is an indication that the design discourse engages reasoning, and produces more design rationale outcomes. In this regard, the reminder card approach promotes more objectivity, analysis and thoughts during a design discourse. However, it does not necessarily assure that the quality of a design is high. In other words, discussing more risks or assumptions does not necessarily mean that the identified risks or assumptions are relevant or important to the final design. Good designers still need to have the knowledge and insights to find those relevant and important risks and assumptions that are meaningful to a design.

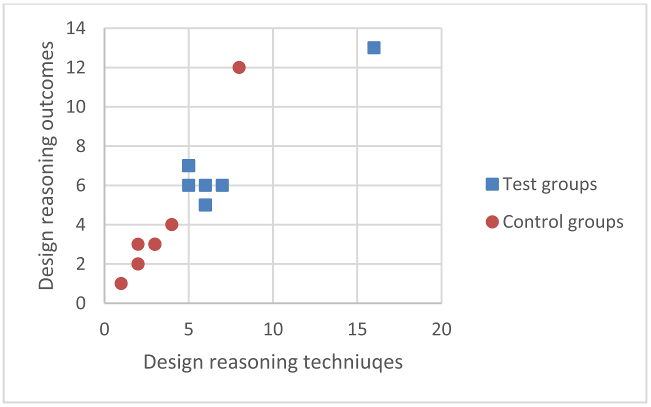


Fig. 11. Scatter plot of the risk analysis technique and the risk design rationale outcome.

5.3. Problem identification and solution option generation

One of the reasoning aspects is to consider more design problems that need solving, and to explore more potential options. With the reminder cards, more discussions on design problem were notable. The amount of improvements of design problem reasoning of the student teams in the test group over the student teams in the control group is 56.7%, whereas the improvements of the professional teams in the test group over teams in the control group is only 7.9% (see Table 6). This could be due to a number of reasons: (a) professionals can naturally think of the design problems without the help of this reminder card; (b) the problem reminder card gives little stimulation on problem thinking for the professionals; or (c) problem discussion is difficult to use. In (Tang and Lau, 2014), it was found that professionals sometimes create solutions without articulating problems. So it seems to point to the fact that the reminder cards as a reminder is not as effective to stimulate problem exploration. Discussions on solution options appear effective for both student and professional teams, with an increase of 74% and 88.7%, respectively (Table 6). In (Tang et al., 2008), we found similarly that by asking the participants to state their design options, junior designers performed better. There was no effect on the more experienced participants. The experiment in (Tang et al., 2008) was to design a user interface. Their exercise was much simpler than the exercise in this experiment. It appears that the use of the Option card in this exercise stimulated much more discussions on options, for both students and professionals. In both experiments, it appeared that a reminder system to the designers to consider options had an effect on option considerations. When a problem is more complex, such a reminder can benefit even the professionals.

We also checked the distinct problems and solution options identified. The average improvements between control and test group for both students and professionals are notable as well. Despite a marginal increase of only 7.9% by the professional teams in the group to explicitly verbalise that problems need to be discussed, the professional teams in the test group produced 84% more distinct problems than their peers in the control group. The student teams in the test group, on the other hand, produced only 34% more distinct problems. Whilst students are reminded of exploring problems, their experience could limit what they can think of as new problems. In terms of solution options, both professionals and student teams in the test group found more options than the teams in the control group. This shows that the reminder card approach has a positive impact on both problem identification and solution option generation.

5.4. Combating satisficing behaviour

Satisficing is a behaviour where a decision maker makes a decision that is good enough to satisfy the goals (Simon, 1987). A satisficing decision maker seeks a somewhat satisfactory solution rather than an optimal one. The reasons for making such decision making are cognitive limitations and time constraints. In software development, designers were found to exhibit satisficing behaviours. Zannier et al. show that designers use this strategy to evaluate if design cues are better or worse, rather than quantifying the right or wrong of design cues (Zannier et al., 2007). Tang and Vliet evaluated designers' satisficing behaviour by the limited amount of reasoning and limited amount of time spent on reasoning before making decision judgments (Tang and van Vliet, 2015).

In the current experiment, we looked at the time that the teams spent on the assignment (Section 4.1 and Appendix A). Although teams in the test group spent on average almost 20% more time on the exercise, the difference between the test and control group is not statistically significant. Furthermore, all teams received specific instructions to spend 2 h on the assignment, and the teams in the test group were instructed to have a reflective moment at 1 h and 45 min, which might have influenced the time taken by the teams as well.

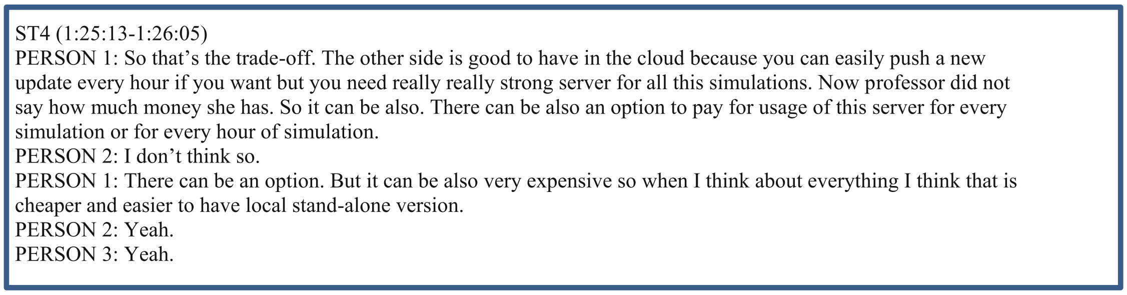

ST4 (1:25:13-1:26:05)
PERSON 1: So that's the trade-off. The other side is good to have in the cloud because you can easily push a new update every hour if you want but you need really really strong server for all this simulations. Now professor did not say how much money she has. So it can be also. There can be also an option to pay for usage of this server for every simulation or for every hour of simulation.
PERSON 2: I don't think so.
PERSON 1: There can be an option. But it can be also very expensive so when I think about everything I think that is cheaper and easier to have local stand-alone version.
PERSON 2: Yeah.
PERSON 3: Yeah.

Fig. 12. Excerpt discussion on Trade-offs.

An interesting finding is that the student teams in the test group used significantly more words in their discussions than the student teams in the control group, on average about 55%. While some people might simply be more verbose, this large and significant difference between the test and control group at least means the teams in the test group had more explicit discussions during which they vocalised their opinions about the design. Furthermore, the average of reasoning techniques applied by student teams in the control group were 33.5 times compared to 56.7 times for the student teams in the test group (Table 3). The reminder cards may have given the teams in the test group more ideas to explore and so there are more discussions. This result plus the rejection of Ha_0 for some reasoning techniques mean that the student teams in the test group used more design reasoning techniques than the student teams in the control group. It infers that the control group reached their design conclusions using less reasoning than the test group. Also, the student teams in the test group found on average 84.4 design reasoning outcomes versus an average of 50.5 for the student teams in the control group (

Table 4). Again combining this with the rejection of Hb_0 for some of the design reasoning outcomes, we can say that the teams in the test group found more design reasoning outcomes than the teams in the control group.

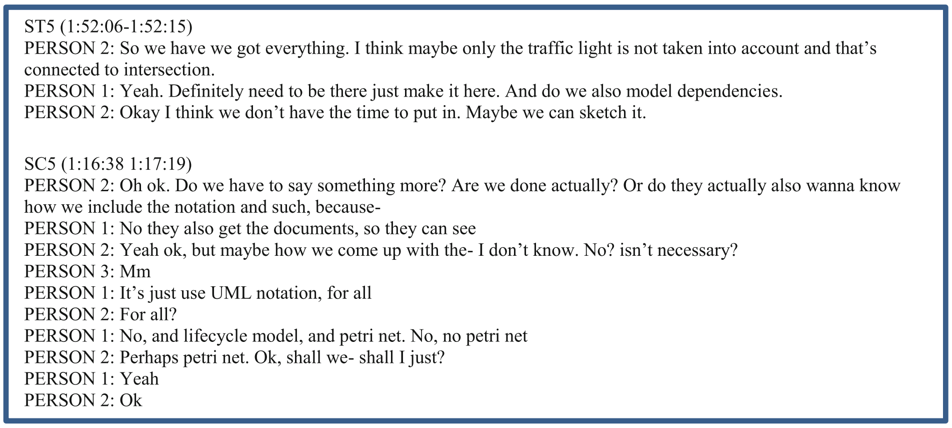
The above results indicate that the reminder card approach – consisting of the reasoning cards and the reflective moments – appears to encourage the designers to reason, discuss and explore their design in more detail. In support, the transcripts show that the teams in the test group exhibit less satisficing behaviour when making decisions. We can see this difference of attitude in the transcripts. These teams mention how they have run out of time before they are completely satisfied with their design. As can be seen in the extract of ST5 (Fig. 13), a new design issue is mentioned, but there is no time to solve it. In contrast, the teams in the control group do not look exhaustively for every potential solution to a problem but typically go with the first solution that is satisfactory. The transcripts show that the teams in the control group, especially those that did not discuss much or reach the two hour mark simply ran out of issues to resolve. An example is illustrated in the extract of SC5 (Fig. 13). Team SC5 was touching on design issues that they needed to solve, but they convinced themselves that what they had was good enough (satisficing). They did not go further into details to explore more about that decision but instead they ended the discussion. These observations show signs that the teams in the control group are more partial to satisficing behaviour.

The comparisons of the number of reasoning statements and the time spent by teams in the control and test group suggest that due to the cards, the test group was reminded to explore the reasoning topics, and systematically explore more about the design. The results support our finding that the reminder card approach leads to applying more reasoning techniques and it combats the natural instincts of satisficing.

5.5. External reflection and reminder cards

This study is based on the concept of reflection developed in (Razavian et al., 2016). Whilst the basis of both studies are the concept of reflection, there are differences between our study approach and the Razavian et al. approach. First, the subjects in our study used the card system to generate their own reflections. The study of external reflection in (Razavian et al., 2016) relied on an external person to prompt the students by asking questions such as "What are your design issues for this service?". Second, the participants of our study were all peers in the design team and anyone can offer a card to start reasoning, and no one played a specialized devil's role. Third, in this study, we assume that all designers have sufficient reasoning capabilities to tackle the given design problem in our experiment, without the need of prior domain knowledge. In the Razavian et al. experiment, the designs were industrial and different for each group. Some students lacked the knowledge and experience to tackle the problems, thus making them more reliant on the external reflector.

The results of the Razavian et al. experiments show that external


ST5 (1:52:06-1:52:15)
PERSON 2: So we have we got everything. I think maybe only the traffic light is not taken into account and that's connected to intersection.
PERSON 1: Yeah. Definitely need to be there just make it here. And do we also model dependencies.
PERSON 2: Okay I think we don't have the time to put in. Maybe we can sketch it.

SC5 (1:16:38 1:17:19)
PERSON 2: Oh ok. Do we have to say something more? Are we done actually? Or do they actually also wanna know how we include the notation and such, because-
PERSON 1: No they also get the documents, so they can see
PERSON 2: Yeah ok, but maybe how we come up with the- I don't know. No? isn't necessary?
PERSON 3: Mm
PERSON 1: It's just use UML notation, for all
PERSON 2: For all?
PERSON 1: No, and lifecycle model, and petri net. No, no petri net
PERSON 2: Perhaps petri net. Ok, shall we- shall I just?
PERSON 1: Yeah
PERSON 2: Ok

Fig. 13. Extracts of transcripts ST5 and SC5: willingness to explore more – satisficing or not.

reflection can help to prompt the students to reason, when they omitted to reason or when they dealt with a problem that they had little domain knowledge. External questions can provide hints. The card system relies on the cards to remind the subjects to reason. Both studies showed that by asking more questions (i.e reminders), more reasoning is performed (i.e. result of reflection) and the quality of the design discourse improves. From the controlled experiment in this study, we further show the extent of improvements in certain kinds of design rationale given the preset design exercise. Furthermore, the simplified reflective questions in terms of cards have positive effects on reasoning, just as an external reflector. So the future research question is “what is the appropriate set of reminder questions to help designers reflect and reason?”. Both studies point out the importance of reminders and reflections in design and both demonstrate that its method can improve design discourse and improve design reasoning.

6. Research validity

6.1. Internal validity

Internal validity is about the validity of the research based on the research setup and interpretation of findings. First, this research makes use of the Irvine assignment which was designed to allow researchers to observe design behaviour (Petre and Van Der Hoek, 2013). The design assignment provides a new and unfamiliar domain such that the designers cannot rely on past experience to solve the design problems. Yet this design assignment is relatively small and cannot represent all kinds of software design. As such, the interpretation of how our subjects carry out this assignment has limitations for its applicability to other kinds of software design assignment. Second, student participants were selected from the software architecture class in the Netherlands. Professional participants in the test group were from Australia, and the professionals in the control group were from the U.S.A. We assume the compatibility of these people culturally and technically. This seems a reasonable assumption. With the professional teams, there were differences such as years of experience, domain knowledge and modelling and design expertise amongst the subjects. These factors may influence the results. With such as assumption, we have found convincing results to show that the reminder card approach had affected our participants in similar ways. This may be because the participants are capable of finding the design rationale, and the main variable is them being mindful of reasoning. We studied the use of the reminder cards on seven different types of reasoning. The experiments provided convincing results to show that the reminder cards had a profound effect generally on the way design reasoning were carried out, for both professionals and students. The questionnaire, as another information source, confirms that our participants agree that the approach has changed the way they think design.

We used discourse analysis on the transcripts. We interpreted what our participants were saying, which in itself is subjective and reliant on the view of the researcher (Horsburgh, 2003). This risk exists but in some ways mitigated when the participants responded positively to the reminder card approach through the questionnaire. Overall, we argue that the risks do not impose a major threat to the interpretation of the evidence that the cards have a positive effect on design reasoning.

6.2. External validity

External validity is about the extent to which the results from a study can be generalized. This study is to experiment with the reminder card approach to test if it makes a difference to how software designers reason. Although the population of our experiment was limited, which threatens generalizability of our findings, the results of our study are in line with previous studies on improving design rationale. A number of similar experiments and studies have been done previously to test if a stimulus of some sort can help designers' to reason more. Van Heesch et al. tested decision documentation (van Heesch et al., 2013). Keil et al. studied the use of checklists on decision making (Keil et al., 2008). Razavian et al. studied the use of reflective questions on design reasoning (Razavian et al., 2016). These studies all pointed out that some external stimulus such as checklists or prompting can help designers to reason more.

Our results did not show a difference between the test and control group on constraints. There may be two explanations. First, a requirement is a natural constraint, our designers were trying to grasp the interpretation of the requirements and what they mean to the design itself. Second, there was not enough time to explore the implementation in enough depth to start exploring constraints in details. As the constraint card did not have much impact on the test group, we cannot conclude its impact on constraint reasoning.

Whilst we have encouraging signs to indicate that the reminder card approach not only improves reasoning discussions, it also increases the identification of design rationale, problem identification and solution options. However, we also saw that experience come into play. Professionals could identify more risks and problems, even though they did not mention that they were doing it. This factor would mean that there are other factors at play when we interpret the results. Unfortunately, for this kind of study, we cannot isolate these influences.

In this work, we do not measure the impact of reasoning on the quality of design. This is because we have no measures to judge if the outcome of one design is better than another. Although based on research such as this one, we may claim that more reasoning can have a positive impact on design discourse, there are other factors such as domain knowledge that influence the end quality of a design. We did not isolate these factors in this study. As shown in (Tang et al., 2008), we are unable to show that teams in the test group would come up with more superior designs than teams in the control group.

6.3. Reliability

Reliability is about ensuring that the results found in the study are consistent, and would be the same if the study is conducted again. To ensure that the coding of the transcripts is reliable it was tested for inter-reliability using Cohens kappa coefficient (Cohen, 1968) to measure the level of agreement between coders. The transcripts were each coded by two researchers using Nvivo 10. The average kappa coefficient of each of the transcripts was above 0.6, which is considered to show a good level of agreement. The average of all transcripts combined is 0.64. We performed Mann–Whitney tests to see if the test and control groups are statistically different from each other. We also used Spearman ranked test to test correlation of events. We recognize that the power of these tests is limited as we only have limited samples.

7. Conclusion and future work

Our research goal was to test the hypothesis if a reminder card approach can improve designers' reasoning and improve the amount of design rationale they can find. We designed the reminder cards based on design reasoning concepts (Razavian et al., 2016). We simplified the reflective questions from (Razavian et al., 2016) and used seven cards to represent the reasoning concepts. This research is an empirical research in the form of an experiment involving an example assignment from the UCI experiment. The assignment was given to both students (12 teams) and professionals (5 teams). There are several dimensions in the analysis: (a) participants cooperated in teams which were divided into a control group and a test group; (b) participants were students and professionals; and (c) analysis was based on the six types of reasoning. We measured the amount of reasoning performed as well as the number of distinct design rationale found.

We observe that the reminder card approach improved reasoning carried out by both students and professionals, and generally enable them to provide more design rationale. The contributions of this study are significant in two ways. First, using the reminder card approach, we have shown that both professionals and students found significantly more design rationale, except for constraints, when designing. The test group spent more time and were more willing to explore the intricacies of the design. The significance of this finding are: (a) A previous study (Tang et al., 2008) found that reminders can help novice but not professionals. This study shows that it can also benefit professionals as well as novice; (b) the reminder cards provide a flexible structure to prompt designers to discuss reasons. This structure appears to have encouraged designers to verbalise their ideas. As a result, the test group exhibits less satisficing behaviour than the control group.

Second, we observed a correlation between the use of design reasoning techniques and the generation of design rationale. We found that the use of reminder cards stimulated the use of reasoning techniques, and that in turn helps to generate more design rationale. This finding shows for the first time how the use of design reasoning techniques can affect the generation of design rationale.

There are a number of things that warrant future investigations. There are differences between the performance of the students and the professionals. For example, student teams in the test group found more distinct design rationales than the professional teams both in the test and control group (Fig. 7). Furthermore, the improvement of the test versus the control group is quite different for students and professionals (Table 6): students perform more problem structuring whereas the improvement of the professionals is more on the other techniques. It would be interesting future research to systematically compare the influence of the reminder card approach on designers with different experience levels. After the experiment, some participants mentioned that it would have been useful to provide guidance on how to use the cards during the design session, perhaps a structured approach or a method to prompt for reasoning and reflection and to guide design discussions may be useful. On the other hand, quite a few participants indicated that the reminder card approach interrupted the natural flow of the discussion. As such, it is worthwhile to further research into how a reminder system and a structured discussion method may help or hinder reasoning in a group design environment.

In the current research, we are interested in how a reminder card approach influences the design reasoning process. What we do not directly examine is whether the amount of explicit reasoning performed by the designers leads to better end-results, i.e. a better overall design. Here it should be noted that explicit design rationale is often seen as an integral part of an architectural design (ISO/IEC/IEEE, 2010), so having a better, more reasoned and explicit design rationale enables designers a better chance, but not a guarantee, of better quality design. A study (Tang et al., 2008) shows that performing explicit design reasoning improves software quality, and we have recently performed a study (de Jong, 2017) in which the designs of architects using guidelines comparable to the reminder cards were consistently rated better by both external experts and peers. These are promising results, but more research is needed to find out how and why any kind of design reasoning methods, techniques or tools would improve the quality of the overall final design.

Appendix A. Time and word count of the design discussions.

Control group	Duration of recording	Word count	Test group	Duration of recording	Word count
SC1	1:43	5197	**ST1**	2:01	14,743
SC2	1:57	7283	**ST2**	1:23	12,893
SC3	1:22	9291	**ST3**	1:59	13,925
SC4	1:13	8791	**ST4**	2:24	14,392
SC5	1:17	7369	**ST5**	1:54	9937
SC6	2:05	11,332	**ST6**	1:51	11,514
PC1	0:59	5805	**PT1**	1:49	13,977
PC2	1:53	12,190	**PT2**	2:00	15,013
PC3	1:52	9969			

Appendix B. Cards played by the test groups.

	ST1	ST2	ST3	ST4	ST5	ST6	PT1	PT2
CONTEXT	1	0	1	0	1	2	1	3
PROBLEM	2	2	0	6	3	5	11	3
SOLUTION	2	2	1	7	3	2	4	1
ASSUMPTION	3	1	2	1	3	2	17	8
CONSTRAINT	1	3	1	0	2	1	10	2
RISK	0	0	1	1	2	4	14	2
TRADE-OFF	0	1	0	1	1	1	10	1
TOTAL	9	9	6	16	15	16	67	20

Appendix C. Terms used by the test groups.

	ST1	ST2	ST3	ST4	ST5	ST6	PT1	PT2
ASSUMPTION	8	12	14	5	12	23	36	42
CONSTRAINT	6	3	1	18	12	21	15	17
RISK	8	19	12	0	5	11	27	4
TRADE-OFF	6	1	9	4	11	20	22	5

Appendix D. Terms used by the control groups.

	SC1	SC2	SC3	SC4	SC5	SC6	PC1	PC2	PC3
ASSUMPTION	0	0	0	3	0	0	6	16	2
CONSTRAINT	0	1	4	5	10	0	0	0	8
RISK	1	0	0	0	0	0	0	0	0
TRADE-OFF	0	0	0	0	0	0	0	0	0

Appendix E. Reasoning techniques applied by the test group.

	ST1	ST2	ST3	ST4	ST5	ST6	PT1	PT2
ASSUMPTION ANALYSIS	14	6	9	5	8	7	10	14
CONSTRAINT ANALYSIS	7	9	2	7	8	10	12	7
RISK ANALYSIS	6	7	6	5	5	5	16	6
TRADE-OFF ANALYSIS	5	2	2	4	2	1	6	2
OPTION GENERATION	19	2	11	6	6	8	14	6
PROBLEM STRUCTURING	33	19	24	25	20	25	22	19

Appendix F. Reasoning techniques applied by the control groups.

	SC1	SC2	SC3	SC4	SC5	SC6	PT1	PT2	PC1
ASSUMPTION ANALYSIS	2	0	2	3	1	3	10	14	5
CONSTRAINT ANALYSIS	4	6	10	7	7	5	12	7	5
RISK ANALYSIS	2	2	3	4	1	3	16	6	3
TRADE-OFF ANALYSIS	1	1	0	3	0	1	6	2	1
OPTION GENERATION	1	3	4	6	9	7	14	6	4
PROBLEM STRUCTURING	15	20	18	12	15	13	22	19	18

Appendix G. Design reasoning outcomes of the test teams.

	ST1	ST2	ST3	ST4	ST5	ST6	PT1	PT2
ASSUMPTION	15	6	6	5	8	7	10	9
CONSTRAINT	8	8	4	7	8	7	9	6
RISK	6	6	5	7	7	6	13	5
PRO	17	4	10	8	4	3	9	4
CON	10	2	8	13	9	6	19	5
DESIGN ISSUES	29	10	17	17	8	13	21	13
DESIGN OPTIONS	42	9	33	28	18	18	43	18
DESIGN DECISIONS	29	10	17	17	8	11	15	12

Appendix H. Design reasoning outcomes of the control groups.

	SC1	SC2	SC3	SC4	SC5	SC6	PC1	PC2	PC3
ASSUMPTION	2	1	2	5	1	3	6	6	2
CONSTRAINT	4	8	9	5	8	8	7	6	10
RISK	2	2	3	4	1	3	2	8	3
PRO	2	4	5	12	3	4	3	7	0
CON	1	2	0	4	2	4	1	5	7
DESIGN ISSUES	3	8	13	19	16	11	9	13	15
DESIGN OPTIONS	5	10	14	18	25	23	11	23	25
DESIGN DECISIONS	4	9	13	20	17	11	8	12	15

References

Abowd, G., Bass, L., Clements, P., Kazman, R., Northrop, L., 1997. Recommended Best Industrial Practice for Software Architecture Evaluation. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Babb, J., Hoda, R., and Norbjerg, J., "Embedding reflection and learning into agile software development," 2014.

Bass, L., Clements, P., Kazman, R., 2012. Software Architecture in Practice, third ed. Addison Wesley, Boston.

Bazeley, P., Jackson, K., 2013. Qualitative Data Analysis With NVivo. Sage Publications Limited.

Bedjeti, A., Lago, P., Lewis, G., de Boer, R.C., Hilliard, R., 2017. Modeling context with an architecture viewpoint. In: Presented at the 1st International Conference on Software Architecture. Gothenburg, Sweden.

Bengtsson, P., Bosch, J., 1998. Scenario-based software architecture reengineering. In: Proceedings of Fifth International Conference on Software Reuse, pp. 308–317.

Boehm, B.W., 1991. Software risk management: principles and practices. IEEE Softw. 8, 32–41.

Børte, K., Ludvigsen, S.R., Mørch, A.I., 2012. The role of social interaction in software effort estimation: unpacking the "magic step" between reasoning and decision-making. Inf. Softw. Technol. 54, 985–996.

Buckingham Shum, S., Hammond, N., 1994. Argumentation-based design rationale: what use at what cost? Int. J. Hum. Comput. Stud. 40, 603–652.

Buckingham Shum, S., Selvin, A.M., Sierhuis, M., Conklin, J., Haley, C.B., Nuseibeh, B., 2006. Hypermedia support for argumentation-based rationale. Rationale Management in Software Engineering. Springer, pp. 111–132.

Bull, C.N., Whittle, J., 2014. Supporting reflective practice in software engineering education through a studio-based approach. IEEE Softw. 31, 44–50.

Burge, J., Brown, D.C., 2000. Reasoning with design rationale. Artificial Intelligence in Design'00. Springer, pp. 611–629.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A., 2016. 10 years of software architecture knowledge management: practice and future. J. Syst. Softw. 116, 191–205.

Carriere, J., 2005. Lightweight Archit. Altern. Assess. Method Available. http://blogs.msdn.com/jeromyc/archive/2005/08/27/457081.aspx.

Cohen, J., 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. Psychol. Bull. 70, 213.

Conklin, J., Begeman, M., 1988. gIBIS: a hypertext tool for exploratory policy discussion. In: Proceedings of the 1988 ACM conference on Computer-supported cooperative work, pp. 140–152.

Conklin, J., Selvin, A., Buckingham Shum, S., Sierhuis, M., 2001. Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In: Proceedings of the 12th ACM conference on Hypertext and Hypermedia, pp. 123–124.

de Jong, P., 2017. Master Research Thesis. Department of Information and Computing Sciences, Utrecht University.

Decisions, S.. (2016, February 26, 2016). *Smart decisions: a software architecture design game*. Available: http://smartdecisionsgame.com/#.

Dorst, K., 2006. Design problems and design paradoxes. Des. Issues 22, 4–17.

Dorst, K., Cross, N., 2001. Creativity in the design space: co-evolution of problem-solution. Des. Stud. 22, 425–437.

Dutoit, A., McCall, R., Mistrik, I., Paech, B. (Eds.), 2006. Rationale Management in Software Engineering. Springer p.^pp. Pages.

Falessi, D., Briand, L.C., Cantone, G., Capilla, R., Kruchten, P., 2013. The value of design rationale information. ACM Trans. Softw. Eng. Methodol. 22, 21.

Grenning, J., 2012. Planning poker or how to avoid analysis paralysis while release planning.–2002. Online. http://renaissancesoftware.net/files/articles/PlanningPoker-v1.

Holyoak, K.J., Simon, D., 1999. Bidirectional reasoning in decision making by constraint satisfaction. J. Exp. Psychol. 128, 3–31.

Horsburgh, D., 2003. Evaluation of qualitative research. J. Clin. Nurs. 12, 307–312.

IDEO, 2015. IDEO method cards. Available. https://www.ideo.com/by-ideo/method-cards/.

ISO/IEC/IEEE, 2010. ISO/IEC/IEEE 42010:2010 Systems and Software Engineering - Architecture Description. Mar.

Kahneman, D., 2003. Maps of bounded rationality: psychology for behavioral economics. Am. Econ. Rev. 93, 1449–1475.

Kahneman, D., Tversky, A., 1972. Subjective probability: a judgment of representativeness. Cognit. Psychol. 3, 430–454.

Kazman, R., Abowd, G., Bass, L., Clements, P., 1996. Scenario-based analysis of software architecture. IEEE Softw. 13, 47–55.

Kazman, R., Asundi, J., Klein, M., 2001. Quantifying the costs and benefits of architectural decisions. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001), pp. 297–306.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J., 1998. The architecture tradeoff analysis method. In: Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '98), pp. 68–78.

Keil, M., Li, L., Mathiassen, L., Zheng, G., 2008. The influence of checklists and roles on software practitioner risk perception and decision-making. J. Syst. Softw. 81, 908–919.

Kollanus, S., Koskinen, J., 2009. Survey of software inspection research. Open Softw. Eng. J. 3, 15–34.

Lago, P., van Vliet, H., 2005. Explicit assumptions enrich architectural models. In: Proceedings 27th International Conference on Software Engineering (ICSE'05), pp. 206–214.

Lamoreux, M., 2005. Improving agile team learning by improving team reflections [agile software development]. In: Agile Conference, 2005. Proceedings, pp. 139–144.

Lockyer, J., Gondocz, S.T., Thivierge, R.L., 2004. Knowledge translation: the role and place of practice reflection. J. Contin. Educ. Health. Prof. 24, 50–56.

Lytra, I., Gaubatz, P., Zdun, U., 2015. Two controlled experiments on model-based architectural decision making. Inf. Softw. Technol. 63, 58–75.

Maclean, A., Young, R., Bellotti, V., Moran, T., 1996. Questions, options and criteria: elements of design space analysis. In: Moran, T., Carroll, J. (Eds.), Design Rationale - Concepts, Techniques, and Use. Lawrence Erlbaum, New Jersey, pp. 53–105.

Moe, N.B., Aurum, A., Dybå, T., 2012. Challenges of shared decision-making: A multiple case study of agile software development. Inf. Softw. Technol. 54, 853–865.

Obbink, H., Kruchten, P., Kozaczynski, W., Postema, H., Ran, A., Dominick, L., et al., 2002. Software Architecture Review and Assessment (SARA) Report (version 1.0).

Petre, M., Van Der Hoek, A., 2013. Software Designers in Action: A Human-Centric Look at Design Work. CRC Press p.^pp. Pages.

Poort, E.R., v. Vliet, H., 2011. Architecting as a risk- and cost management discipline. In: Proceedings of the Ninth IEEE/IFIP Working Conference on Software Architecture, pp. 2–11.

Razavian, M., Tang, A., Capilla, R., Lago, P., 2016. In two minds: how reflections influence software design thinking. J. Softw. 28, 394–426.

Reymen, I., 2001. Improving Design Processes through Structured Reflection: A Domain-independent Approach,". PhD Thesis. Stan Ackermans Institute, Centre for Technological Design, Technische Universiteit Eindhoven, Eindhoven.

Rittel, H.W.J., Webber, M.M., 1973. Dilemmas in a general theory of planning. Policy Sci. 4, 155–169.

Schön, D.A., 1983. The Reflective practitioner: How Professionals Think in Action. EUA: Basic Books, Nueva York.

Schriek, C., 2016. Master Research Thesis. Department of Information and Computing Sciences, Utrecht University.

Schriek, C., van der Werf, J.M.E.M., Tang, A., Bex, F., 2016. Software architecture design reasoning: a card game to help novice designers. In: Tekinerdogan, B., Zdun, U., Babar, A. (Eds.), Software Architecture: 10th European Conference, ECSA 2016, Copenhagen, Denmark, November 28 – December 2, 2016, Proceedings. Springer International Publishing, Cham, pp. 22–38.

Schwaber, K., Beedle, M., 2002. Agile Software Development With Scrum 1 Prentice Hall, Upper Saddle River.

Simon, H.A., 1987. Satisficing. The New Palgrave: A Dictionary of Economics 4. pp. 243–245.

Stacy, W., MacMillan, J., 1995. Cognitive bias in software engineering. Commun. ACM 38, 57–63.

Sweller, J., 1988. Cognitive load during problem solving: effects on learning. Cogn. Sci. 12, 257–285.

Tang, A., Aleti, A., Burge, J., van Vliet, H., 2010. What makes software design effective? Des. Stud. 31, 614–640.

Tang, A., Lau, M.F., 2014. Software architecture review by association. J. Syst. Softw. 88, 87–101 2//.

Tang, A., Tran, M.H., Han, J., van Vliet, H., 2008. Design reasoning improves software design quality. In: Proceedings of the Quality of Software-Architectures (QoSA 2008), pp. 28–42.

Tang, A., van Vliet, H., 2015. In: Weyns, D., Mirandola, R., Crnkovic, I. (Eds.), Software Designers Satisfice," in Software Architecture 9278. Springer International Publishing, pp. 105–120.

Tversky, A., Kahneman, D., 1986. Rational choice and the framing of decisions. J. Bus. 59, S251–S278.

Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. IEEE Softw. 22, 19–27.

Van Heesch, U., Avgeriou, P., Hilliard, R., 2012. Forces on architecture decisions-a viewpoint. In: Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on, pp. 101–110.

van Heesch, U., Avgeriou, P., Tang, A., 2013. Does decision documentation help junior designers rationalize their decisions?-A comparative multiple-case study. J. Syst. Softw. 86, 1545–1565.

Van Manen, M., 1977. Linking ways of knowing with ways of being practical. Curriculum Inq. 6, 205–228.

van Vliet, H., Tang, A., 2016. Decision making in software architecture. J. Syst. Softw. 117, 638–644.

Vliet, Hv., 2008. Software Engineering: Principles and Practice, third ed. John Wiley & Sons.

Von Alan, R.H., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28, 75–105.

Wiltschnig, S., Christensen, B.T., Ball, L.J., 2013. Collaborative problem–solution co-evolution in creative design. Des. Stud. 34, 515–542.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A., 2012. Experimentation in Software Engineering. Springer Publishing Company Incorporated.

Woods, E., 2012. Industrial architectural assessment using TARA. J. Syst. Softw. 85, 2034–2047.

Zalewski, A., Borowa, K., Ratkowski, A., 2017. On cognitive biases in architecture decision making. In: Lopes, A., de Lemos, R. (Eds.), *Software Architecture: 11th European Conference, ECSA 2017*, Canterbury, UK, September 11–15, 2017, Proceedings. Springer International Publishing, Cham, pp. 123–137.

Zannier, C., Chiasson, M., Maurer, F., 2007. A model of design decision making based on empirical results of interviews with software designers. Inf. Softw. Technol. 49, 637–653.

Zimmermann, O., Koehler, J., Leymann, F., Polley, R., Schuster, N., 2009. Managing architectural decision models with dependency relations, integrity constraints, and production rules. J. Syst. Softw. 82, 1249–1267.

Antony Tang is Associate Professor in Swinburne University of Technology, Australia. He received a PhD degree in Information Technology from Swinburne in 2007. Prior to being a researcher, he had spent many years designing and developing software systems. His main research interests are software architecture design reasoning, software development processes, software architecture knowledge management.

Floris Bex is a researcher and instructor at the Utrecht University, The Netherlands.

Courtney Schriek graduated with a degree of Master Science in Business Informatics from Utrecht University. She is currently a consultant at Deloitte, The Netherlands.

Jan Martijn van der Werf is assistant professor in Software Architecture at the Department of Information and Computing Science at Utrecht University. His research focuses on behavioral aspects within software architectures, such as the design and verification of component-based systems, process modelling, and Petri nets.