# Software Designers Satisfice

Antony Tang[1(✉)] and Hans van Vliet[2]

[1] Swinburne University of Technology, Melbourne, Australia
atang@swin.edu.au
[2] VU University, Amsterdam, The Netherlands
hans@cs.vu.nl

**Abstract.** The software architecture community has advocated design rationale in the last decade. However, there is little knowledge of how much reasoning is performed when software design judgments are made. In this study, we investigated the amount of design reasoning performed before making a decision. We recruited 32 students and 40 professionals to participate in this software architecture design study. We found that most subjects needed only a few reasons before making their decisions. They considered that giving a few reasons were good enough to judge despite that more reasons could be found. This result shows a satisficing behavior in design decision making. We explore the implications of this common behavior on software architecture design.

**Keywords:** Satisficing · Design judgment · Design reasoning · Behavioral software engineering

## 1 Introduction

Software teams often employ software engineering methodologies and processes during development. While the use of methodologies and processes can help, they do not ensure that good design decisions are made. Curtis suggested that the impact of behavior on software productivity is greater than the use of tools and methods [1]. As Donald Schön eloquently said: "We [professionals] are bound to an epistemology of [design] practice which leaves us at a loss to explain, or even to describe, the competences to which we now give overriding importance" [2]. His observation that we cannot explain how a designer thinks is still an accurate description of the state of software design. Software architecture design as a series of design decisions and design rationale have been recognized by many in the software architecture community [3, 4]. Many studies and methods promoted the capture of design rationale [5, 6]. However, we know little of how much reasoning is performed to create design rationale for making decisions. This issue has become clear in a recent workshop on software design [7]. The amount of reasoning software designers perform is an important issue to explore as the presence of design rationale is insufficient to ensure that a design decision is well reasoned.

Software architecture design is a complex business. Architects and designers often face uncertainties as they explore the problem space and the solution space. Decisions


D. Weyns et al. (Eds.): ECSA 2015, LNCS 9278, pp. 105–120, 2015.
DOI: 10.1007/978-3-319-23727-5_9

are made and reasoning is performed with many unknowns. Simon argues that due to bounded rationality, i.e. the limitations of our cognitive ability, the idea of maximisation in design is untenable. Realistically, we cannot consider all possible design options to achieve an optimal design. As such, complete design exploration by traversing the entire problem and solution spaces is impossible. Instead, such design approach should be replaced by the idea of statisficing [8]. Satisficing indicates that a decision maker makes a decision that is good enough to satisfy the goals [9].

If design optimization is not possible, how much reasoning and explorations do designers do before making decisions? How much satisficing is good enough? On the other hand, we know that missing information can potentially cause design flaws [10, 11]. In order to gain some insights on these questions, we set up an experiment to test how professionals and students reason and judge. In this study, we found no significant differences, in terms of the number of reasons and the judgments, between the students and professionals. We found that our subjects typically stopped reasoning and made judgment after finding few reasons.

In order to understand why both groups provided so few reasons, we did a follow up study and asked professional participants when and why they stop reasoning. They said that they chose to stop as soon as they were convinced that they had enough reasons. This is a satisficing behavior. However, a minority group of professionals are the exceptions to this reasoning behavior. We call them non-satisficing designers. They reasoned more thoroughly than most of the students and professionals. They had a different way of reasoning and we found three judgment characteristics.

## 2 Software Design Judgment

Design judgment and decision making involves reasoning, but other psychological elements such as motivation and cognitive limitation also influence reasoning activities. Software architecture design methodologies typically assume that software designers reason rationally. That is, if a process or a standard is followed and rationale is provided, the quality of the rationale is not questioned [12, 13]. Several studies, however, have found that software designers behave opportunistically [14, 15]. These studies show software designers do not use systematic reasoning to arrive at a design. In this study, we explore software design reasoning from different perspectives:

- How much reasoning do designers give before making design judgments? Why?
- Do professionals and students reason differently?
- What are the characteristics of designers' reasoning behavior?

To study these aspects of design decision making, we prepared vignettes, or scenarios, to describe software system scenarios. Reasoning with scenarios is one of the things that software designers do, and more so for software architects. Software designers and architects are often presented with functional and quality requirements, use cases, system goals and other information. From this information, they have to explore the problem space, reason with the situations and synthesize solutions [16]. In this study, each scenario describes a software system, its requirements and context

at a high level. A conclusion is provided to allow the participants to reason with the scenario. The conclusion is worded in a controversial way to provoke reasoning and argument. For example, in one scenario (*scenario 5*), we described a proposed smart-card ticketing system to be built in Pakistan. We provided a controversial conclusion that says: "*It is viable to implement this system in Pakistan*". The participants were asked to do two things. First, they were asked to provide written reasons for disagreeing or agreeing with the given conclusion. The controversial conclusions were worded such that many reasons can be found to object them. The number of reasons, the nature of the reasons and the judgments given by the participants on software design scenarios allowed us to study how a participant uses reasoning and make judgments with software scenarios.

Second, the participants were asked to indicate their level of agreement with the conclusion. A seven-point Likert-like scale was used to indicate their level of agreement. As the conclusions are controversial and deliberately contain many arguments against them, the experiment setup allows us to examine the relationships between reasoning and disagreement conviction. We expected issues to be found in each scenario, and we wanted to observe what reasons the participants would give. The data also allowed us to analyse the given reasons, and relate them with their judgments, i.e. the level of disagreement. An online survey tool was used to gather the data.

### 2.1 Research Approach

In this research, we provided design scenarios as stimuli to gather designers' responses. The use of vignette to do this kind of study is a valid approach in the fields of social and cognitive psychology [17]. For our research goals, this research approach is appropriate as it would allow us to measure reasoning responses of participants using the same stimuli that are relevant to software architecture design.

In the experiment, we asked our participants to write down those reasons, to simulate the process stipulated by many rational design methods where designers are required to provide design rationale. We collected the data and encoded all unique reasons given. We counted those reasons and performed statistical analysis. We first tested if professionals would produce more reasons to support their design judgments than students. Second, we investigated their design judgements to compare how much disagreement they had with the conclusions. Third, we used a questionnaire to find out how some of our professionals carried out the exercises.

We did two rounds of study, with 61 participants in the first round comprising 32 students and 29 professionals. They were asked to do the exercises only. In the second round, the 11 participants had to do the exercises and then they were questioned about the process of their reasoning as well. When analysing the data from both rounds, we took a constructivist approach [12] in which we interpreted the data collected to build a theory of how software designers reason.

### 2.2 Scenario Preparation

We developed ten scenarios based on some actual system development cases. We worded the scenarios to allow issues to be embedded within the scenarios. Due to the controversial conclusions, we anticipated that there would be many reasons to challenge the conclusions, especially in complex cases. We wanted to discover what reasons the participants would find. We devised the scenarios such that possible reasons would be typical of the types of issues identified by Meyer [8]. For each scenario, there is a vignette that describes the scenario. A conclusion is given to stipulate reasoning and argumentation. The participants were asked to judge the conclusions and give reasons/issues for why they (dis)agree. Ten scenarios were developed and six of them were chosen to be used in the study. From these six scenarios, two were considered common (scenario 1 and 10) in that it was possible that the professionals had had a chance to encounter them in the work place; the other four scenarios (scenario 5, 6, 7 and 9) were complex and uncommon in that few people would have had the chance to work with such systems. This means that it is highly likely that these scenarios were new, in terms of making design judgment, to most participants. Uncommon and complex scenarios mean that the participants had to carefully question the scenarios [14].

We used an online survey to show the vignettes and the conclusions. The online survey tool gathered the reasons and the level of agreement given. There was no time limit on how long the participants could take to complete the online survey.

### 2.3 Pilot Testing

We invited four people to participate in our trials. They were given all ten scenarios. We found ambiguous wording in some scenarios and they were rectified. Out of the ten scenarios, we chose six scenarios for the actual study because we found that these six scenarios already contained all the different types of issues that we might find, and the pilot participants were able to identify them.

### 2.4 Participants' Demographics

We invited students and software professionals to participate. The first group contained 32 second year Bachelor Computer Science students from the Web-systems Project at VU University Amsterdam. There were 40 software professional participants, 29 in the first round and 11 in the second round. We used availability and snowballing sampling methods to recruit them. Many of the industry participants were either known to the researchers through work-related contacts or were colleagues of software developers known to the researchers. There were 6 architects, 1 academic, 16 software engineers and designers, 6 in IT and software management, 4 analysts, 6 consultants and 1 Database Administrator. This group of software professionals had an average industry experience of 16 years.

### 2.5 Data Gathering

We analysed the reasons given by the participants and identified unique reasons in each scenario to count them. We call this collection of reasons for each scenario a normative set of reasons. These reasons were typically assumptions, constraints and risks stated to argue against the conclusions. Analogies were also used to reason for or against a conclusion. We call them For-Analogy and Against-Analogy. In the second round, we used a questionnaire to get more information from the participants after they had completed the exercise. The purpose hereof was to find out how they reasoned and when they stopped reasoning.

## 3 Results and Analysis

### 3.1 Limited Amount of Reasoning by Students and Professionals

First, we analyzed the total number of issues provided for all six scenarios. Students on average found 6.56 issues for all 6 cases taken together, and professionals found 7.55 issues for the 6 cases. We performed an independent sample t-test to compare the mean issues found between the two groups and the results showed insignificant difference. In conclusion, professionals on average did not provide more reasons than students when they made their design judgments.

Second, the number of issues given by both the students and the professionals in each scenario were low as compared with the normative set. The number of issues given, as a percentage of the normative set, was between 9% to 13% in complex scenarios, and about 18% for scenario 10, and 30% (students) and 40% (professionals) in scenario 1. Scenario 1 has only two issues in the normative set. In most cases, although many reasons could have been given, only a small number of reasons were.

Third, participants found different reasons. Some participants said that they found the important reasons and stopped. However, there are many other valid reasons and it is somewhat subjective and arbitrary to argue that some reasons are more *important* than others. Let us examine the possible reasons for Scenario 5 (see vignettes and normative set of reasons[1]). In Scenario 5, the vignette is "*A new contactless smart-card system is designed to be used in the public transport in Pakistan. Each traveller would need to pay a deposit to obtain a personalized smart-card. Each card costs US$4.50. With the smart-card, a passenger can travel on all public transport system such as bus, train and mini-bus throughout the country. The smart-card can be reloaded at ATM machines or over the counter at a bank. This system will replace all cash tickets in 18 month*". The conclusion suggests that "*It is viable to implement this system in Pakistan*". Issue 1 challenges the affordability of a travel card in a poor country. Issue 1 was found by 6 students and 13 professionals. Issue 2 challenges the viability of building the infrastructure. Identification of this issue requires system construction experience. It was found by 2 students and 12 professionals. Issue 6

[1] http://www.ict.swin.edu.au/personal/atang/documents/Design%20Reasoning%20Experiment-V1.5.pdf

states the limitation of the people having access to bank accounts and ATMs. Issue 6 was found by 11 students and 15 professionals. Many students and professionals found these three popular issues. Other issues were not commonly cited. It is difficult for the researchers to say which issue is more important. However, using this example, we see that all the issues are equally valid but participants found different ones.

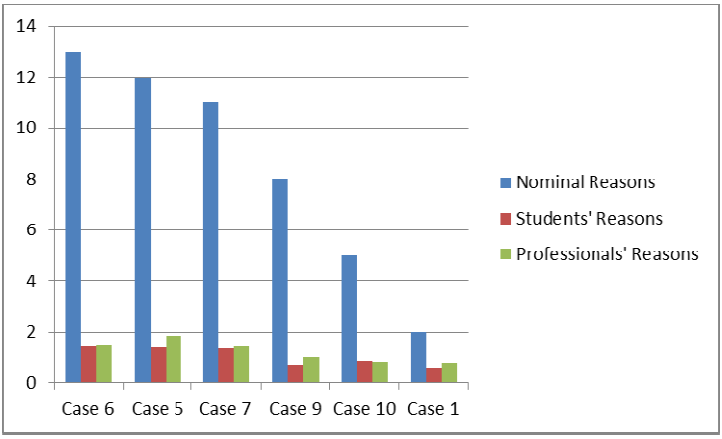


**Fig. 1.** Average Number of Reasons Identified by Student and Professional Groups Compared with the Total Number of Nominal Reasons

If these are valid issues, why did most participants not continue to find more of them before making their judgments? Table 1 shows the average number of reasons and as a percentage of the normative set of reasons found by the students and the professionals. Both students and professionals found only a fraction of the issues except for the simpler Case 1. A viable explanation was that they had been using the least cognitive efforts [18]. These professionals, and all the students, found some reasons and they decided that these reasons were *good enough* to make their judgments. This is an indication of their satisficing behavior. We verified this result by asking why they stopped reasoning (see Section 3.4).

**Table 1.** Average Reasons Identified and As A Proportion of Normative Set

| | All Cases | Case 1 | Case 5 | Case 6 | Case 7 | Case 9 | Case 10 |
|---|---|---|---|---|---|---|---|
| Normative Set of Reasons | 51 | 2 | 12 | 13 | 11 | 8 | 5 |
| Students Average Reasons | 6.56 (12.8%) | 0.59 (29.6%) | 1.46 (12.2%) | 1.5 (11.5%) | 1.37 (12.5%) | 0.71 (8.9%) | 0.9 (18.1%) |
| Professional Average Reasons | 7.55 (14.8%) | 0.8 (40%) | 1.86 (15.4%) | 1.52 (11.7%) | 1.47 (13.4%) | 1.05 (13.1%) | 0.85 (17%) |

### 3.2 Judgment Conviction

We expected that professionals having more experience than students would have been more convicted about their disagreement. This is because they could see the many issues in the given conclusions. Instead, the professionals did not totally object the conclusions. They objected to the conclusions in three cases and they were not totally convicted to the disagreement (see Table 2).

The two groups had very similar level of agreement in all scenarios. We performed a non-parametric test of significance on the median of agreement between the two groups and found no statistical difference between them in any of the cases. As discussed earlier, the level of agreement and the number of reasons of both groups in each of the cases shows no statistical difference. Thus, the two groups judge these cases in the same way. They found similar number of reasons to support their decisions and made similar judgments.

**Table 2.** Median Level of Agreement

| | Case 1 | Case 5 | Case 6 | Case 7 | Case 9 | Case 10 |
|---|---|---|---|---|---|---|
| Students : Median Level Agrmnt | 4.0 | 2.0 | 2.0 | 4.0 | 5.0 | 5.0 |
| Professionals: Median Level Agrmnt | 2.0 | 2.0 | 2.0 | 5.0 | 4.5 | 5.0 |

### 3.3 Amount of Time Spent on Reasoning

We tested the difference in time spent between students and professionals using Independent Sample t-Test. We found that there is no significant difference in the time spent by the students and the professionals in any of the cases (see Table 3).

**Table 3.** Average Time Spent by Subjects

| | Case 1 | Case 5 | Case 6 | Case 7 | Case 9 | Case 10 |
|---|---|---|---|---|---|---|
| Students Avg Time | 4.68m | 4.87m | 5.03m | 4.43m | 4.68m | 4.68m |
| Professionals Avg Time | 4.23m | 4.23m | 4.41m | 3.70m | 4.46m | 3.84m |

Almost all of the participants did not have hands-on experience with any of the scenarios. This precludes the chance that someone could reason with the scenarios based on intimate working domain knowledge. Despite the lack of domain knowledge, participants only spent a small amount of time on reasoning and judging. This could mean that reasoning is a function of time. After our participants spent a certain amount of time which they consider as *enough* and found what they thought were *reasonable* reasons to judge, they stopped.

### 3.4 Questionnaire Results

Following the first round of experiments, we found that professionals reasoned similarly to students. They stopped reasoning after finding few reasons. They had similar

judgments with the students and the judgments were non-convicting despite that more reasons could have been found. The professionals also spent a similar amount of time as the students. In order to understand these phenomena, we invited more professionals to participate in this experiment in the second round. In addition to the design reasoning exercises, these professionals had to answer some questions afterwards. In the questionnaire, we asked them if they had an urge to complete the task quickly. The median in a 7-point Likert scale is 5, indicating a tendency wanting to complete the task quickly. We asked the professionals if they wanted to complete the task thoroughly. The median in a 7-point Likert scale is 5.5, meaning they wanted to do a thorough job.

We checked if any of the professionals had hands-on experienced in the domain in any of the cases. There was only one developer who did some work with Case 10. There were no domain familiarities with any of the other cases by any professionals. This helps to reduce the risk that hands-on development experience and prior domain knowledge might have biased the results in that judgments are based on prior knowledge rather than explicit reasoning. We asked the professionals to estimate the number of possible reasons for each case. The results were mixed. For professionals who had guessed that there are more reasons, why did they not find more of them? The following are some of the answers they gave.

- "I stopped when I found the main reasons. I did not need to find more reasons."
- "I drew a line at the time I spent in each case."
- "Gut feeling, based on experience and projects I have seen."

We asked the professionals if they were to find more reasons, would they have been more disagreeable with the conclusions. The following are the answers they gave.

- "YES. I would need to do more analysis. If I did, I would come up with more reasons. But when I saw a red flag, that was enough for me to disagree."
- "POTENTIALLY YES. If I spent more time, I would come up with more information to argue against the conclusion."
- "YES. I would have gone into more depth. The reasons I gave were important."
- "YES. To explore more techniques and to measure their feasibility according to risk and trade-offs."

These answers basically show that (a) the professionals knew that they could have found more reasons; (b) they stopped because *there is no need*; (c) they said they would have been more disagreeable had they found more reasons.

## 4 Satisficing Behavior

Software design is said to be a wicked problem [19] because it is complex and the situations that a designer faces are often new and unfamiliar. In a complex environment, the solutions, the problems and the reasoning behind them are not obvious. We

designed the scenarios to be complex and so they require careful reasoning. The results of this study show that (a) students and professionals provided few reasons before judging; (b) students and professionals spent a similar amount of time before judging; (c) the level of agreement of students and professionals on the conclusions was similar and their judgments were non-convicting; (d) some professionals said that they could have found more reasons but they stopped because they thought it was enough. Given these results, we examined a number of theories to explain our results.

### 4.1 Explaining the Results

First, designers might behave in an opportunistic way when designing [15]. Instead of analysing a design systematically and thoroughly, they use information that is readily available. This behavior could be due to the limited cognitive capacity of designers to process all requirements and design information simultaneously, but there is no extra information from our study to support the theory of cognitive overload. Comments by some participants partially supported the opportunistic behavior, like "*solution that pops up in the mind*".

Second, psychologists have suggested two modes of thinking: System 1 and System 2. System 1 operates automatically and quickly with little effort. System 2 allocates attention and effortful mental activities. In one experiment, it was shown that more than 50% of Harvard, MIT and Princeton students used intuitive thinking to solve a problem but they gave the wrong answer [20]. Other studies have shown that as people become more skilled in a certain task, the mental efforts they spend on that task are reduced. The *Law of Least Effort* asserts that people will gravitate to the least demanding course to accomplish the same goals. This law applies to cognitive efforts as well, it shows that people are inclined to use less of System 2, even in a situation that demands analysis [18]. In our experiment, we cannot show which system our participants used. But there were hints, based on the comments made by our participants, that participants engaged in both intuition (System 1 thinking) and effortful analysis (System 2 thinking).

Klein suggests "we need to blend systematic analysis and intuition. Neither gives us a direct path to the truth. Each has its limitations" [20]. Hammond suggests that judgments is an exercise to cope with uncertainty [21], and if judgment is a rivalry between intuition and analysis, then if a person's uncertainty is somewhat satisfied, by intuition or by analysis or both, the cognitive process stops and a judgment is made. Zannier et al. found a similar blend of Rational Decision Making (RDM) and Naturalistic Decision Making (NDM) when studying software designers [14]. These two theories point to the same decision making process: intuition vs. analysis. Our participants told us that they did some analysis, but they also used their intuition.

Third, Simon suggested that it is not possible to find an *optimal solution*. Instead, designers design systems incrementally and if a design appears to work, that design is selected and a designer moves on. Therefore, designers do not maximise or optimise a design. S/he chooses a good enough design. Simon suggested that maximisation is untenable and should be replaced by the idea of satisficing. That is, we cannot have a perfect solution, but instead we have a good enough solution [8, 9].

Finding as many reasons as one can achieve is maximization or optimization behavior. Our participants clearly opted to find just-enough reasons before judging. Although there is enough evidence to demonstrate designers are *satisficed* with their judgments, we do not know why *satisfaction* occurred, whether it was due to using System 1 thinking, or a laziness in System 2 thinking, or bounded rationality, or cognitive overloading. With the establishment of the satisficing behavior of software designers and software architects, we examine its implications on software architecting.

### 4.2 Satisficing in Software Architecture Design

Using the results that we gathered, we triangulate the evidence to conclude that software designers use satisficing when making decisions. We summarise our arguments below and discuss their implications:

**Student and Professionals Satisfice to the Same Degree.** There is no statistically significant difference to how much students or professionals reason or when they stop finding more reasons. This shows that, irrespective of experience, software designers are satisficed at similar points. *Implications*: As satisficing is a natural thing for most software designers, it is important to recognize such and train software architects and designers to investigate a problem deeper when dealing with complex issues.

**Satisficing and Time Bounded Decision Making.** All professionals who participated in the second round said they had an urge to complete the tasks quickly, even though they wanted to do a thorough analysis. We don't know where that urge to complete the tasks came from. A number of these professionals commented that they could have spent more time but they did not. *Implications*: This result seems to indicate that satisficing behavior and time bounded decision making are intertwined. Judgments are made quickly because of finding good-enough reasons or a designer has judged that enough time has been spent. Ward interprets satisficing as "choosing among a subset of behavior when information processing or time constraints limit the ability of a decision maker to make an optimal decision" [22]. There is no explicit time constraint in the experiment, but the designers implicitly considered time as a factor. Time is shown to be a cost factor in the study of purchasing behavior [23]. As such, time is like cost, it is spent in order to gain something. In software decision making, the perception of how much time spent on a problem becomes a cost function that is a trade-off with the potential gains of spending that time.

**Satisficed Reasoning.** Many issues had been identified in the normative set of issues in the experimental scenarios. However, most participants typically found few of them before judging. Out of the 51 unique issues identified for all 6 scenarios, students found an average of 6.5 and professionals found an average of 7.5 issues. Some participants commented they had found the main reasons and so they stopped, and yet they were not convicted to their judgments. This seems to indicate they knew they did not have all the reasons to make judgments but they thought that it was good-enough. *Implications*: The potential issue of such satisficing behavior is that the resulting judgments based on partial reasoning may be incomplete and flawed. As software

designers often face new and unfamiliar situations, like the scenarios provided in the experiments, and if these situations require careful analysis in order to consider the many intricate and interrelated scenarios and requirements, then the reasoning before judgment is inadequate. If the result of this study is a general reflection of the way software designers make decisions, then there is a potential software design thinking issue to consider. Satisficing may work fine when a designer is experienced and familiar with the domain, but it may be risky in complex and unfamiliar design situations that require thorough analysis [24]. The question is how we may recognize satisficing, and that it may be causing design risks?

## 5 Non-Satisficing Professionals

In the student group, the number of issues found was evenly distributed, and there were no outliers (see Figure 2). However, in the professional group, we discovered a sub-group of 4 professionals who reasoned differently. These are the outliers who appear not to use satisficing judgments. We contrast these non-satisficing professionals with satisficing software designers by comparing the following evidence: number of reasons given, judgment conviction, time spent and the use of analogy. The existence of this group of designers is a contrast to the satisficing designers, and the ways they differ help to characterize satisficing behaviors.

### 5.1 Non-Satisficing Professionals Reason More

Non-satisficing professionals cited many more reasons than the other two groups. They found almost double the number of reasons. They also considered a broader context of the design situations to challenge the given conclusions. The reasons that these professionals identified were not very obvious and they could not have been found without careful thinking. They were also more thorough in their analysis.

As shown in Fig. 2, the average number of reasons this non-satisficing group identified is double that of the student group and the rest of the professionals. We ran an independent samples Kruskal-Wallis test to compare the number of reasons found between the three groups. We found that the non-satisficing group found significantly more reasons than the other groups ($p=0.004$).

We further analysed these four professionals in the non-satisficing group for each of the six scenarios, and found that they were clearly outliers in every scenario. In each scenario, the average number of reasons they found was typically double that of the students and the other professionals. This result highlighted that they reason different from the rest of their peers and the students. We found statistical significant difference between this group and all the other participants. However, with such a small number, the power of the statistical tests is limited.

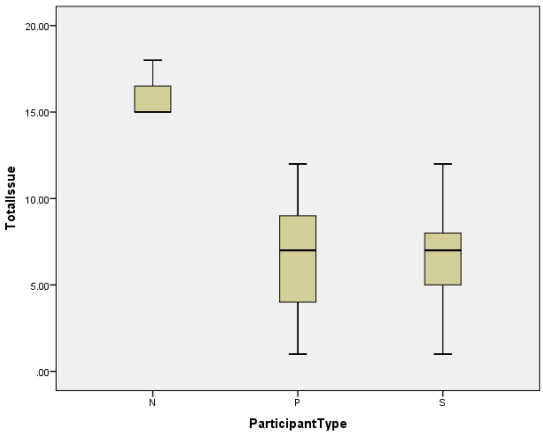


**Fig. 2.** Reasons Identification by Non-Satisficing Group (N), Satisficing Professional Group (P) and Student Group (S)

### 5.2 Non-Satisficing Professionals Use Less Analogy

Examining the reasons given by the participants, we noticed that analogies were often used to support or refute the conclusions. Typical arguments were "*XXX is successful, so it should also be applicable to YYY*". But often software design is situated within particular contexts [25], so analogy as an argument would work only when the situations of the cases are very similar.

Analogies are patterns that are used to simplify learning [26]. It is an intuitive way to allow matching solutions to problems that are similar. The use of analogies by students and some professionals suggest that intuitions were used in their design thinking style. Non-satisficing professionals appeared to use far fewer analogies than the other two groups. Table 4 summarizes the number of reasons by analogies given by the three groups. The number in each cell indicates (a) the number of analogies given; and (b) the average number of analogies (within brackets) given by the participants in a group. Let us examine a couple of example analogies found by the participants in Case 5: "*There are some good examples of smart-card systems implemented in other countries' public transport. I don't see why there should be any more difficulties in a country such as Pakistan*." and "*The description of the system is very similar to London's Oyster card. However, Pakistan's economy is less developed than that of Great Britain. It is very likely that equipment will fail to work after a while and so travellers will be forced to find ways to circumvent the ticket barriers*."

The analogies given were somewhat relevant to the reasoning and argument of the Pakistani situation. However, one could point out many differences between Pakistan, London and The Netherlands. If we examine them we see they are not very good reasons. Non-satisficing professionals hardly used analogy in their argumentation. When they used analogy, they gave reasons to support them. On the other hand, the other two groups used more analogies on average in 4 different scenarios. Many were not substantiated by sound arguments. The use of unsupported analogy is a characteristic of satisficing reasoning.

**Table 4.** Analogies per Participant (analogy counts and averages within brackets)

| | Case 5 | Case 7 | Case 9 | Case 10 |
|---|---|---|---|---|
| Students | 12 (0.375) | 8 (0.25) | 1 (0.03) | 1 (0.03) |
| Satisficing Professionals | 6 (0.24) | 7 (0.28) | 0 (0) | 2 (0.08) |
| Non-Satisficing Professionals | 2 (0.25) | 0 (0) | 0 (0) | 0 (0) |

### 5.3 Non-Satisficing Professionals are More Convicted to their Judgments and Spend More Time Reasoning

We note that the non-satisficing professionals are more convicted to their judgment. This contrasts with the average results reported in Table 2. We argue that non-satisficing professionals disagreed more because they found more reasons to support their arguments and refuted the conclusions. They were also more confident with their judgments. Dörner studied the habits of good and bad decision makers. One observation he made was that a good decision maker asks many why questions to explore reasons, challenges the assumptions, and poses different scenarios [27]. Non-satisficing professionals have these traits as well. They identified more issues and were more certain of their judgments.

The non-satisficing professionals spent significantly more time than the other professionals (Table 5). The cost of time did not appear to affect them as they focused on the reasoning tasks. In every case, they spent more time and found more reasons to support their judgment conviction.

**Table 5.** Average Time Spent by Professionals

| | Case 1 | Case 5 | Case 6 | Case 7 | Case 9 | Case 10 |
|---|---|---|---|---|---|---|
| Satisficing Professionals | 3.87m | 3.76m | 4.07m | 3.44m | 4.21m | 3.64m |
| Non-Satisficing Professionals | 6.75m | 7.5m | 6.75m | 5.5m | 6.25m | 5.25m |

## 6 Threats to Validity

There are a number of limitations in our study. One might argue that, in a real-life situation, one would do more reasoning but many of the professionals we surveyed after the experiment said that they stopped because they felt that there were enough reasons to convince them (Section 3.4). The results obtained here could reflect how these participants reason and judge.

Second, one potential construct validity issue is that the participants might not want to spend a lot of time to work on the scenarios, and inadvertently limited the amount of reasons given. However, many professionals indicated they stopped because they felt they had identified enough reasons, and did not say that they ran out of time.

Third, this experiment had a limited number of scenarios. The scenarios are short and the participants knew that they were dealing with an experiment albeit that they did not know its purpose. This is potentially a construct validity issue and limits our

ability to generalize the results. We only conducted the survey with the second round of 11 participants, the number of responses was limited and as such it limits what we can generalize from these responses. Fourth, the way vignette is used in this study is similar to how information is communicated with software designers although in real-life there are often opportunities for further questioning, and other environmental factors such as time-allowance and presence of expertise that may influence designer reasoning behavior. As such, we cannot claim general representativeness.

Finally, we interpret the experimental results using the theory of satisficing. In the experimental construct and the interpretation of the results, we assume that (a) the kind of reasoning embedded in the scenarios are close to real-life; (b) there is little familiarity of our participants with the domains; (c) our participants are motivated to provide us good information. In these regards, the evidence that we gathered appear to be consistent based on the results that we collected: the reasons given, the judgments and the time-spent. The comments made by the second round of professionals clearly point to satisficing behavior. We found an outlier group, and use their behavior to contrast the behavior exhibited by the student group and most of the professionals.

## 7 Conclusions

In this work, we study how much reasoning designers do when they are asked to provide design rationale in different design scenarios. We asked designers to provide design reasons before making design judgements. We had 72 participants in our experiment. We provided six scenarios for them to reason with. The results show that most students and professionals provided few reasons before they make a judgment. Their judgments were non-convicting, i.e. they did not totally disagree with the conclusions. They told us that when they found enough reasons, they stopped looking for more reasons and made their judgments. This result shows general satisficing behavior in design reasoning. A small portion of professionals are non-satisficing designers. We identify three characteristics that contrast with the behaviour of satisficing designers: (a) these professionals seldom used analogy; (b) they provided twice as many reasons than the others before judging; (c) when they judged, they were more convicted to their judgments and willing to spend more time to reason before judging.

This study has shown that software designers and architects use satisficing in judging design scenarios. If this practice is representative of the everyday practice of software architecting, then it has significant implications. The software architecture community generally assumes that the presence of rationale is good enough to improve design quality. This study shows that designers only provide a fraction of design rationale. The amount of reasoning and the level of their satisficing could impact on the quality of design decisions, especially in unfamiliar design situations [24]. In a different situation, satisficing could work if the design complexity is low. As such, it is important to recognize satisficing behavior and its potential risks to design quality. Potential solutions to address the issue of premature satisficing lie in the recognition of this human behavior during design. Reflective design thinking [28], better design time management, reasoning techniques [29], architecture design and analysis

techniques that include recognition and critical appraisal of satisficing behavior, debiasing [30] and managing design complexities [24] are some of the related approaches that could improve software design practice, and all of them need further exploration.

**Acknowledgments.** We thank Patricia Lago for her initial contribution to this work. We thank Alice Yip for encoding the data. We also thank our participants.

## References

1. Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems. Commun. ACM **31**, 1268–1287 (1988)
2. Schön, D.A.: The reflective practitioner : how professionals think in action. Basic Books, Nueva York (1983)
3. Falessi, D., Briand, L.C., Cantone, G., Capilla, R., Kruchten, P.: The value of design rationale information. ACM Transactions on Software Engineering and Methodology (TOSEM) **22**, 21 (2013)
4. Jansen, A., Bosch, J.: Software architecture as a set of architectural design decisions. In: Proceedings 5th IEEE/IFIP Working Conference on Software Architecture, pp. 109–120 (2005)
5. Dutoit, A., McCall, R., Mistrik, I., Paech, B. (eds.): Rationale Management in Software Engineering. Springer (2006)
6. Tang, A., Barbar, M.A., Gorton, I., Han, J.: A Survey of Architecture Design Rationale. Swinburne University of Technology (2005)
7. Petre, M., Van Der Hoek, A. (eds.): Software Designers in Action: A Human-Centric Look at Design Work. CRC Press (2013)
8. Simon, H.A.: The Sciences of the Artificial. The MIT Press (1996)
9. Simon, H.A.: Satisficing. The New Palgrave: a Dictionary of Economics **4**, 243–245 (1987)
10. Meyer, B.: On Formalism in Specifications. IEEE Software **2**, 6–26 (1985)
11. Tang, A., Lau, M.F.: Software architecture review by association. Journal of Systems and Software **88**, 87–101 (2014)
12. Perry, D.E., Wolf, A.L.: Foundation for the Study of Software Architecture. ACM SIGSOFT Software Engineering Notes **17**, 40–52 (1992)
13. ISO/IEC/IEEE: ISO/IEC/IEEE 42010:2010 Systems and software engineering - Architecture description, March 2010
14. Zannier, C., Chiasson, M., Maurer, F.: A model of design decision making based on empirical results of interviews with software designers. Information and Software Technology **49**, 637–653 (2007)
15. Guindon, R.: Designing the design process: exploiting opportunistic thoughts. Hum.-Comput. Interact. **5**, 305–344 (1990)
16. Kruchten, P.: What do software architects really do? Journal of Systems and Software **81**, 2413–2416 (2008)
17. Tversky, A., Kahneman, D.: The framing of decisions and the psychology of choice. Science **211**, 453–458 (1981)
18. Kahneman, D.: Thinking, fast and slow. Penguin (2011)
19. Rittel, H.W.J., Webber, M.M.: Dilemmas in a general theory of planning. Policy Sciences **4**, 155–169 (1973)

20. Klein, G.: Streetlights and shadows: Searching for the keys to adaptive decision making. The MIT Press (2009)
21. Hammond, K.R.: Human judgement and social policy: Irreducible uncertainty, inevitable error, unavoidable injustice. Oxford University Press (1996)
22. Ward, D.: The role of satisficing in foraging theory. Oikos **63**, 312–317 (1992)
23. Fasolo, B., Carmeci, F.A., Misuraca, R.: The effect of choice complexity on perception of time spent choosing: When choice takes longer but feels shorter. Psychology & Marketing **26**, 213–228 (2009)
24. Tang, A., van Vliet, H.: Design Strategy and Software Design Effectiveness. IEEE Software **29**, 51–55 (2012)
25. Gero, J.S., Kannengiesser, U.: The situated function–behaviour–structure framework. Design Studies **25**, 373–391 (2004)
26. Clement, J.: Using bridging analogies and anchoring intuitions to deal with students' preconceptions in physics. Journal of Research in Science Teaching **30**, 1241–1257 (1993)
27. Dörner, D.: The Logic Of Failure: Recognizing And Avoiding Error In Complex Situations. Basic Books (1996)
28. Babb, J., Hoda, R., Norbjerg, J.: Embedding Reflection and Learning into Agile Software Development (2014)
29. Tang, A., Lago, P.: Notes on Design Reasoning Techniques (V1.4). Swinburne University of Technology (2010)
30. Kahneman, D., Lovallo, D., Sibony, O.: Before you make that big decision. Harvard Business Review **89**, 50–60 (2011)