WARSAW UNIVERSITY OF TECHNOLOGY

DISCIPLINE OF SCIENCE: Information and Communications Technology

FIELD OD OF SCIENCE : Engineering and technology

Ph.D. Thesis

Klara Maria Borowa, M.Sc.

Cognitive Biases in Architectural Decision-Making: Impact and Debiasing Strategies

Supervisor

D.Sc. Eng. Andrzej Zalewski

WARSAW 2024

Acknowledgments

This thesis is the fruit of a long journey. A journey that would never have been possible without the many people who supported me. Some directly influenced this research, and some simply helped me out on a personal level. There is no possibility of listing them all, but I will attempt to do so nonetheless.

First and foremost, I want to thank my family wholeheartedly. My husband Piotr, who did all the hard husbandly things, comforted me when I was sad, pushed me to do better, and took care of our kids when I needed to work. He also helped find some participants for this research and distributed many questionnaires for me. I would like to thank my children, Łucja and Maksymilian, for simply being here - you are two bright spots of pure love in my life. Also, special thanks to my mom, who supported me from the beginning until now.

Secondly, I would like to thank all researchers, more or less experienced, who took part in and co-authored my research. This includes two of my colleagues from my team: Andrzej Ratkowski and Szymon Kijas. Also, numerous master's students from Warsaw University of Technology who took part in my research at various points: Robert Dwornik, Sebastian Kamoda, Piotr Ogrodnik, Maria Jarek, Gabriela Mystkowska, Weronika Paszko, Rafał Lewanczyk, Klaudia Stpiczyńska and Patryk Stradomski. My friend Adam Saczko, who was my gate to video game development, and Damian Kowalski, who provided his psychology expertise.

Thirdly, I would like to thank two researchers who believed in this research and performed the final empirical validation experiments in their counties: Marion Wiese and Rodrigo Rebouças de Almeida. Working with you both was a pleasure!

Last but not least, I would like to thank my supervisor, Andrzej Zalewski. This journey started 7 years ago when he recommended Daniel Kahneman's book "Thinking, Fast and Slow". None of this research would have ever happened without that spark that has grown into a nice bonfire. This was quite the journey. Thank you for being here for it all!

Cognitive Biases in Architectural Decision-Making: Impact and Debiasing Strategies

Abstract.

Context: Cognitive biases are natural heuristics impacting each person's decision-making process. Biased decisions cannot be considered fully rational since they are based on various oversimplifications and mental shortcuts. Cognitive biases have been researched in numerous software engineering subdomains, such as testing, requirements engineering, and software effort estimation. In most cases, the biases' were a source of various faulty decisions, and as such, researchers strive to alleviate the biases' influence on software practitioners. The impact of cognitive biases has been of particular interest in the domain of software architecture since the architecture itself can be viewed as a set of design decisions. The impact of cognitive biases on software architecture can be severe, e.g., it may result in incurring unnecessary architectural technical debt. As such, the software architecture community has recognized that research on debiasing methods for software architects is a pressing necessity.

Goal: This thesis presents a set of studies that firstly explored rationales behind architectural decisions and how cognitive biases impact architectural decision-making. Since this impact was proven to be a danger to software projects, the secondary goal of debiasing (i.e., alleviating the impact of biases) was defined. As such, this thesis answers the following research questions:

- RQ1: What rationales are the main reasons behind decisions impacting software practitioner's architectural decision-making?
- RQ2: How do cognitive biases impact architectural decision-making?
- RQ3: Does cognitive biases' impact on architectural decision-making cause architectural technical debt?
- RQ4: How can the negative impact of cognitive biases on architectural decision-making be alleviated?

Method: In order to answer the research questions, various research methods were utilized. The data on rationales behind architectural decisions (RQ1) was obtained through questionnaires and interviews with software practitioners. Information about specific biases impacting architectural decision-making (RQ2) was gathered during a group workshop with software practitioners. Specifics about cognitive biases' influence on incurring architectural technical debt (RQ3) were explored through a series of semi-structured interviews with architects. The debiasing intervention (RQ4) was designed and tested in increments: first through a small pilot study with two groups of students, then through a controlled experiment with 12 groups of students, and then finally verified in an experiment with experienced practitioners.

Results: This thesis' results are numerous. Firstly, the most often used rationales behind architectural decisions were identified, as well as the understanding of why practitioners use these rationales (RQ1). Secondly, a list of 11 cognitive biases that have an impact on architectural decision-making, as well as their consequences, was created (RQ2). The impact of cognitive biases on incurring architectural technical debt was found, which resulted in the finding that three dominating biases are the most problematic: optimism bias, confirmation bias, and anchoring (RQ3). Finally, a debiasing workshop was designed and empirically proven to be beneficial for both students and experienced practitioners (RQ4).

Conclusions: The series of studies presented in this paper resulted in a deepened understanding of cognitive biases on architectural decision-making. Since this impact appeared to be potentially severely dangerous, a debiasing treatment to alleviate the impact of biases was designed and empirically validated. Materials allowing the replication of the debasing intervention and the debiasing workshop are available online.

Keywords: Cognitive bias, Software Architecture, Architectural decision-making, Debiasing

Błędy Poznawcze w Podejmowaniu Decyzji Architektonicznych: Wpływ oraz Przeciwdziałanie

Streszczenie.

Kontekst: Błędy poznawcze to naturalne heurystyki, które wpływają na proces podejmowania decyzji przez każdego człowieka. Decyzje podjęte pod ich wpływem, nie mogą być uważane za w pełni racjonalne, ponieważ opierają się na różnych uproszczeniach i skrótach myślowych. Błędy poznawcze były badane w wielu poddziedzinach inżynierii oprogramowania, takich jak testowanie, inżynieria wymagań i szacowanie nakładu pracy potrzebnego do realizacji projektów informatycznych. W większości przypadków błędny poznawcze były źródłem różnych opacnzych decyzji, w związku z czym badacze starają się złagodzić wpływ uprzedzeń na praktyków oprogramowania. Wpływ błędów poznawczych jest szczególnie znaczący w dziedzinie architektury oprogramowania, ponieważ sama architektura może być zdefiniowana jako zbiór decyzji. Wpływ błędów poznawczych na architekturę oprogramowania może być dotkliwy, na przykład mogą wpłynąć na zaciągnięcie niepotrzebnego architektonicznego długu technicznego. W związku z tym społeczność badaczy architektury oprogramowania uznaje że badania nad metodami przeciwdziałania błędom poznawczym dla architektów oprogramowania są pilną koniecznością.

Cel: Niniejsza rozprawa przedstawia zestaw badań, które miały na celu: (1) Umożliwienie odkrycia powszechnie używanych kryteriów podajmowania decyzji architektonicznych oraz (2) Zbadnie wpływu błędów poznawczych na podejmowanie decyzji architektonicznych. Ponieważ wpływ ten okazał się potencjalnym zagrożeniem dla projektów informatycznych, zdefiniowano drugorzędny cel badań, czyli złagodzenie wpływu błędów poznawczych. W związku z tym niniejsza rozprawa odpowiada na następujące pytania badawcze:

- RQ1: Jakie powody stoją za podejmowaniem decyzji architektonicznych przez praktyków oprogramowania?
- RQ2: W jaki sposób błędy poznawcze wpływają na podejmowanie decyzji architektonicznych?
- RQ3: Czy wpływ błędów poznawczych na podejmowanie decyzji architektonicznych jest powodem zaciągania architektonicznego długu technicznego?
- RQ4: W jaki sposób można złagodzić negatywny wpływ błędów poznawczych na podejmowanie decyzji architektonicznych?

Metoda: Aby odpowiedzieć na pytania badawcze, wykorzystano różne metody badawcze. Dane dotyczące racjonalności decyzji architektonicznych (RQ1) uzyskano za pomocą kwestionariuszy i wywiadów z ekspertami. Informacje na temat konkretnych błędów poznawczych wpływających na podejmowanie decyzji architektonicznych (RQ2) zostały zebrane podczas warsztatów grupowych z ekspertami. Szczegółowe informacje na temat wpływu błędów poznawczych na zaciąganie architektonicznego długu technicznego (RQ3) zostały zbadane poprzez serię wywiadów z architektami oprgramowania. Interwencja przeciwdziałająca (RQ4) została zaprojektowana i przetestowana w trzech kropah: najpierw poprzez małe badanie pilotażowe z dwiema grupami studentów, następnie poprzez eksperyment z 12 grupami studentów, a następnie ostatecznie zweryfikowana w eksperymencie z doświadczonymi praktykami.

Wyniki: Wyniki tej pracy są różnorodne. Po pierwsze, znaleziono najczęściej używane powody stjące za decyzjami architektonicznymi, a także odkryto, dlaczego praktycy uznają te powody za ważne (RQ1). Po drugie, stworzono listę 11 błędów poznawczych, które mają wpływ na podejmowanie decyzji architektonicznych, a także ich konsekwencje (RQ2). Ustalono wpływ błędów poznawczych na zaciąganie architecktoniczneg długu technicznego, w wyniku czego stwierdzono, że trzy dominujące błędy poznawcze są najbardziej problematyczne: optymizm, potwierdzenie i zakotwiczenie (RQ3). Ostatecznie, zaprojektowano warsztat przeciwdziałający, które okazay się korzystny zarówno dla studentów, jak i doświadczonych praktyków (RQ4).

Wnioski: Seria badań przedstawionych w niniejszej rozprawie zaowocowała pogłębionym zrozumieniem błędów poznawczych w podejmowaniu decyzji architektonicznych. Ponieważ wpływ ten okazał się potencjalnie bardzo niebezpieczny, zaprojektowano i empirycznie zwalidowano strategię łagodzącą wpływ błędów poznawczych. Materiały pozwalające na replikację tej interwencji, czyli warsztatu przeciwdziałającego, są dostępne online.

Słowa kluczowe: Błąd poznawczy, Architektura Systemów Informatycznych, Podejmowanie decyzji architektonicznych, Przeciwdziałanie błędom poznawczym

Contents

Acknowledgments . . . 3

1. Introduction . . . 15

1.1. Research Questions . . . 17

1.2. Author's published work . . . 18

1.3. Thesis Outline . . . 20

2. State of the Art . . . 24

2.1. Cognitive biases . . . 24

2.2. Cognitive biases in software engineering . . . 27

2.2.1. Cognitive biases in requirements engineering . . . 28

2.2.2. Cognitive biases in implementation . . . 28

2.2.3. Cognitive biases in testing . . . 29

2.2.4. Cognitive biases in software effort estimation . . . 30

2.2.5. Other impacts of cognitive biases on software engineering activities . . . 31

2.3. Cognitive biases and architectural decision-making . . . 31

2.4. Debiasing . . . 32

3. What rationales drive architectural decisions? An empirical inquiry 34

3.1. Preface . . . 35

3.2. Abstract . . . 36

3.3. Introduction . . . 36

3.4. Related Work . . . 37

3.5. Method . . . 38

3.5.1. Questionnaire: data-gathering . . . 38

3.5.2. Questionnaire: analysis . . . 39

3.5.3. Interviews: data gathering . . . 41

3.5.4. Interviews: analysis . . . 42

3.6. Results . . . 42

3.6.1. RQ1 & RQ2: Most frequent rationales and prioritised software quality attributes . . . 42

3.6.2. RQ3: Rationales' origins . . . 46

3.7. Discussion . . . 49

3.8. Threats to validity 51
3.9. Conclusion 51

4. On Cognitive Biases in Architecture Decision Making 53

4.1. Preface 54
4.2. Abstract 55
4.3. Introduction 55
4.4. Related Work 56
4.5. Investigating Biases in Architectural Decision Making 57
4.5.1. Workshop on Biases in Architecture Decision-Making 57
4.5.2. Influence of Cognitive Biases on Architecture Decision-Making 58
4.6. Cognitive Biases in the Practical Conditions of Architectural Decision Making . 67
4.7. Results 68
4.8. Discussion, Limitations 69
4.9. Summary and Research Outlook 70

5. The Influence of Cognitive Biases on Architectural Technical Debt . 72

5.1. Preface 73
5.2. Abstract 74
5.3. Introduction 74
5.4. Related Work 75
5.5. Research Method 76
5.5.1. Cognitive biases 76
5.5.2. Architectural Debt items 77
5.5.3. Research procedure 78
5.5.4. Study participants 79
5.5.5. Analysis Procedure 80
5.6. Results 81
5.6.1. Architectural debt items influenced by cognitive biases 81
5.6.2. Cognitive biases that influence ATD items 82
5.6.3. Influence of cognitive biases on ATD items 83
5.6.4. Cognitive bias antecedents (RQ4) 88
5.6.5. Possible debiasing methods (RQ5) 92
5.7. Discussion 93

5.8. Threats to Validity 94
5.9. Conclusion and Research outlook 94

6. Is knowledge the key? An experiment on debiasing architectural decision-making - a pilot study 96

6.1. Preface 97
6.2. Abstract 98
6.3. Introduction 98
6.4. Related Work 99
6.5. Study Design 99
6.5.1. Bias selection 99
6.5.2. Data acquisition 100
6.5.3. Data Analysis 100
6.5.4. Participants 101
6.6. Results 101
6.7. Threats to validity 103
6.8. Discussion 103
6.9. Research outlook 104
6.10.Conclusion 105

7. Debiasing architectural decision-making: a workshop-based training approach 106

7.1. Preface 107
7.2. Abstract 108
7.3. Introduction 108
7.4. Related work 109
7.5. Research Method 110
7.5.1. Biases 110
7.5.2. Architectural decision-making task 111
7.5.3. Debiasing Workshop Design 111
7.5.4. Sample 112
7.5.5. Analysis 112
7.6. Results 112
7.6.1. Arguments. 113

7.6.2. Decisions. 113
7.6.3. Cognitive biases. 113
7.6.4. Debiasing techniques. 114
7.7. Discussion 114
7.8. Threats to Validity 115
7.9. Conclusion and Future Work 115

8. Debiasing Experts 117

8.1. Preface 118
8.2. Abstract 120
8.3. Introduction 120
8.4. Related work 122
8.4.1. Cognitive biases 122
8.4.2. Debiasing 124
8.5. Method 124
8.5.1. Sample 125
8.5.2. The experiment 126
8.5.3. Debiasing Workshop 127
8.5.4. Data Analysis 128
8.6. Results 130
8.6.1. Statistical significance 131
8.6.2. Arguments and Counterarguments 133
8.6.3. Cognitive biases 134
8.6.4. Debiasing techniques 134
8.6.5. Decisions 134
8.7. Discussion 135
8.7.1. Lessons learned 136
8.8. Threats to validity 137
8.9. Conclusion 139
8.10.Data availability 139

9. Discussion and limitations 140

9.0.1. Limitations 145

10.Conclusion 146

10.0.1. Data availability 147
References 149
List of Figures 166
List of Tables 167

1. Introduction

Software engineering is a field originally conceived to foster the application of scientific knowledge in various phases of software development, with a particular focus on software design, software construction, and writing technical documentation [1]. The first conference, which focused on topics related to software engineering, was organized by NATO in 1968 [2], where scientists focused on software design, production, and maintenance methods. Currently, the definition of software engineering has been expanded into: "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" [3].

Software architecture is a subdomain of software engineering, which was recognized as an extremely significant part of software design already during the second software engineering NATO conference in 1969 [4]. Software architecture itself can be defined in various ways, e.g.: (1) a set of software elements and their interactions [5] or (2) a set of architectural design-decisions [6]. There are various ways of describing a system's architecture, such as through the 4+1 views model [7], the ISO/IEC/IEEE 42010 standard [8], and the C4 model [9].

Cognitive bias is a term created by psychology researchers Amos Tversky and Daniel Kahneman [10]. According to them, cognitive biases are systematic errors that arise from the heuristics of the human mind that are used unconsciously to simplify judgmental operations [10]. These simplifications are an inherent feature of the human mind, and they are extremely useful in performing mundane, low-stakes, everyday decisions, such as deciding which route to take in order to travel home or what to eat for breakfast. Without heuristics simplifying decisions, the human body would be excessively worn down since entirely rational logic-based thinking uses the body's resources, e.g., it lowers blood glucose concentration [11]. However, this feature can become an impediment in the case of critical high-stakes decisions. For example, investors may prefer to buy stocks of companies that have a short, easy-to-pronounce name [12]. The discoveries related to understanding how human psychology decision-making had been the basis on which Daniel Kahneman obtained the Nobel Prize in Economy in 2002 [13].

Software engineering research on cognitive biases is quite diverse since cognitive biases can impact every aspect of human judgment, including all decisions made by practitioners in the software development process [14]. Notable software engineering disciplines researched in the context of biases include requirements engineering [15], task time estimation [16], agile software development [17], testing [18], architecture [19] and implementation [20].

Cognitive biases in software architecture are an important research area [21] since software architecture can be defined as a set of design decisions [6] and cognitive biases may distorted the decision-making process. Research so far indicates that software architects are often biased [22] and that the biases influence may result in subpar architectural solutions [23]. Such biased decisions may have various negative consequences, such as incurring dangerous [24] architectural technical debt [25], choosing novel solutions without proper risk assessment [26], or assuming that technologies which the development team used previously are always the best choice [22]. As such, understanding the possible negative impact of cognitive biases on software architecture and alleviating this effect is an important research challenge.

The ultimate goal of this research is to create a debiasing strategy that would reduce the negative impact of cognitive biases on architectural decisions, which would improve quality of architectures designed by software practitioners.

The **contributions** comprise a series of solutions to the research problems (defined through the research questions). The contributions, which are all grounded in empirical data, can be summarised as follows:

- Establishing rationales that practitioners use to justify their architectural decisions (Chapter 3).
- Defining a list of cognitive biases impacting architectural decision-making, as well as their consequences (Chapter 4).
- Finding which cognitive biases may cause the occurrence of architectural technical debt. The consequences of such technical debt and the cognitive biases antecedents were also found (Chapter 5).
- Discovering "the wicked triad," an interaction between three major cognitive biases (anchoring, optimism bias, and confirmation bias). This triad seems to be a major reason behind numerous biased architectural decisions (Chapter 6).
- The design and empirical validation of a successful debiasing workshop, which was validated on students (Chapters 6 and 7) and practitioners (Chapter 8). The importance of this contribution stems from the fact that **no previous research has produced an empirically proven debiasing strategy for architectural decision-making**.
- The discovery that debiasing interventions may impact students and practitioners differently. Students' improvement is noticeable since they formulate more non-biased (i.e., fact-based) arguments for and against architectural solutions. Yet, the number of bias occurrences does

not decrease (see Chapter 7). In turn, Practitioners' improvement manifests in fewer bias occurrences and fewer biased arguments in favor of architectural solutions (see Chapter 8). This implies that debiasing interventions must be adjusted to the participant's experience level to be effective.

This thesis' contributions are beneficial for both software engineering researchers and practitioners.

Researchers obtain knowledge about how architectural decisions are made in practice: what rationales motivate practitioners while making decisions, which cognitive biases distort architectural decisions, and how these biases interact with each other. Furthermore, they are now equipped with an empirically proven effective debiasing intervention. This knowledge can be used to develop future methods (e.g., debiasing intervention) and guidelines (e.g., the use of debiasing techniques)for enhanced architectural decision-making.

Practitioners may directly use the teaching materials prepared for the debiasing intervention as a basis for performing their own debiasing workshop in their workplace.

Overall, the knowledge obtained through this study is a step towards better software quality. This can be achieved by making software architects less impacted by cognitive biases, hence making more rational, informed decisions.

1.1. Research Questions

The **research problems** this thesis aims to address are expressed as a set of research questions. The main focus of the research presented in this is the impact of cognitive biases on architectural decision-making. The focal point of this research is finding which cognitive bias influences are negative and how their impact can be reduced.

To explore this issue in depth, this thesis aims to answer the following research questions:

- **RQ1: What rationales are the main reasons behind decisions impacting software practitioner's architectural decision-making?**

  The purpose of this research was to understand which factors practitioners consider most frequently during the architectural decision-making process. Some of the discovered rationales may be cognitive bias antecedents. For example, previous positive experiences with a specific framework may be an antecedent for the anchoring bias [10]. When impacted by anchoring, practitioners may choose a framework based on their previous experiences with it instead of the fit for the project's conditions.

- **RQ2: How do cognitive biases impact architectural decision-making?**
  This research aimed at identifying the cognitive biases that influence architectural decision-making. Additional focus was put into finding the consequences of these biases. Such a list could be used in subsequent studies as a set of reference biases to be researched.
- **RQ3: Does cognitive biases' impact on architectural decision-making cause architectural technical debt?**
  Incurring technical debt is synonymous with making a decision that favors short-term benefits over long-term ones [27]. Architectural technical debt, in particular, is considered to be dangerous because of its severe consequences [24] and lack of adequate refactoring strategies that would make it possible to repay such debt [28]. As such, the aim of this research was to evaluate whether decisions to incur architectural technical debt may be influenced by cognitive biases.
- **RQ4: How can the negative impact of cognitive biases on architectural decision-making be alleviated?**
  Having established that cognitive biases may have a negative impact on architectural decision-making and its results, the subsequent research goal was the creation of an effective debiasing treatment. Previous research suggests a general lack of such debiasing treatments in the software engineering domain [14]. However, creating such seemed possible since a successful debiasing treatment was previously designed for debiasing software effort estimation [29].

In this thesis, Research Questions are used instead of a Thesis since this view on the research matter allows for methods more open to finding additional, unexpected (yet still valuable) insights. A particular example is Chapter 6, which contains an article about a failed debiasing attempt. Despite failing to debias the participants, this paper's findings were crucial in developing a successful debiasing strategy.

1.2. Author's published work

This thesis is based on a set of five published papers, as well as one yet unpublished experiment. The papers included in this thesis as chapters are:

1. Andrzej Zalewski, **Klara Borowa**, and Andrzej Ratkowski. "On cognitive biases in architecture decision making." Software Architecture: 11th European Conference, ECSA 2017, Canterbury, UK, September 11-15, 2017, Proceedings 11. Springer International

Publishing, 2017. (2024 MNiSW points: 140)
Thesis author's contribution: data gathering, data analysis, writing the paper's first draft as well as making changes to the final paper.

2. **Klara Borowa**, Andrzej Zalewski, and Szymon Kijas. "The influence of cognitive biases on architectural technical debt." 2021 IEEE 18th International Conference on Software Architecture (ICSA). IEEE, 2021. (2024 MNiSW points: 140)
Thesis author's contribution: original research idea, research method design, data gathering, half of the data analysis (coding), most of the paper writing.
3. **Klara Borowa**, Robert Dwornik, and Andrzej Zalewski. "Is knowledge the key? an experiment on debiasing architectural decision-making-a Pilot study." Product-Focused Software Process Improvement: 22nd International Conference, PROFES 2021, Turin, Italy, November 26, 2021, Proceedings 22. Springer International Publishing, 2021. (2024 MNiSW points: 70)
Thesis author's contribution: original research idea, research method design, data gathering, half of the data analysis (coding), most of the paper writing.
4. **Klara Borowa**, Maria Jarek, Gabriela Mystkowska, Weronika Paszko, and Andrzej Zalewski "Debiasing architectural decision-making: a workshop-based training approach." European Conference on Software Architecture. Cham: Springer International Publishing, 2022. (2024 MNiSW points: 140)
Thesis author's contribution: original research idea, research method design, data gathering, half of the data analysis (coding), most of the paper writing.
5. **Klara Borowa**, Rafał Lewanczyk, and Klaudia Stpiczyńska, Patryk Stradomski, and Zalewski, Andrzej. "What rationales drive architectural decisions? An empirical inquiry." European Conference on Software Architecture. Cham: Springer Nature Switzerland, 2023. (2024 MNiSW points: 140)
Thesis author's contribution: original research idea, research method design, questionnaire analysis most of the paper writing.

Other published works from the author:

1. Andrzej Zalewski, **Klara Borowa**, and Damian Kowalski. "On cognitive biases in requirements elicitation." Integrating research and practice in software engineering (2020): 111-123. (2024 MNiSW points: 20)
2. **Klara Borowa**, Andrzej Zalewski, and Adam Saczko. "Living With Technical Debt—A

Perspective From the Video Game Industry." IEEE Software 38.6 (2021): 65-70. (2024 MNiSW points: 100)

3. Andrzej Zalewski, **Klara Borowa**, and Krzysztof Lisocki. "Supporting Architectural Decision-Making with Data Retrieved from Online Communities." International Conference on Dependability and Complex Systems. Cham: Springer International Publishing, 2021. (2024 MNiSW points: 20)
4. Szymon Kijas, and **Klara Borowa**. "Evolution Process for SOA Systems as a Part of the MAD4SOA Methodology." International Conference on Dependability and Complex Systems. Cham: Springer International Publishing, 2021. (2024 MNiSW points: 20)
5. **Klara Borowa**, Sebastian Kamoda, Piotr Ogrodnik, Andrzej Zalewski. "Fixations in Agile Software Development Teams." Foundations of Computing and Decision Sciences 48.1 (2023): 3-18. (2024 MNiSW points: 40)
6. Wiese, Marion, and **Klara Borowa**. "IT managers' perspective on Technical Debt Management." Journal of Systems and Software 202 (2023): 111700. (2024 MNiSW points: 100)

1.3. Thesis Outline

This thesis is divided into ten chapters. Figure 1.1 gives an overview of the outline, as well as what published work is contained in particular chapters.

The overview is explained below:

- **Chapter 1** is an introduction to the topic, containing not only an introduction to the research area and motivation behind this work but also information on the author's published work, the contributions of this thesis as well as a precise definition of the original solution to the research problem.
- **Chapter 2** contains the information about the related work and state of the art.
- **RQ1: What rationales are the main reasons behind decisions impacting software practitioner's architectural decision-making?** is answered as part of **Chapter 3**. This chapter presents an empirical inquiry through a questionnaire and a set of semi-structured interviews, which resulted in a list of rationales that practitioners reported using during architectural decision-making. The reasons why these rationales are common, as well as the differences between rationales used by practitioners with different experience levels, are explained in detail in this paper as well.

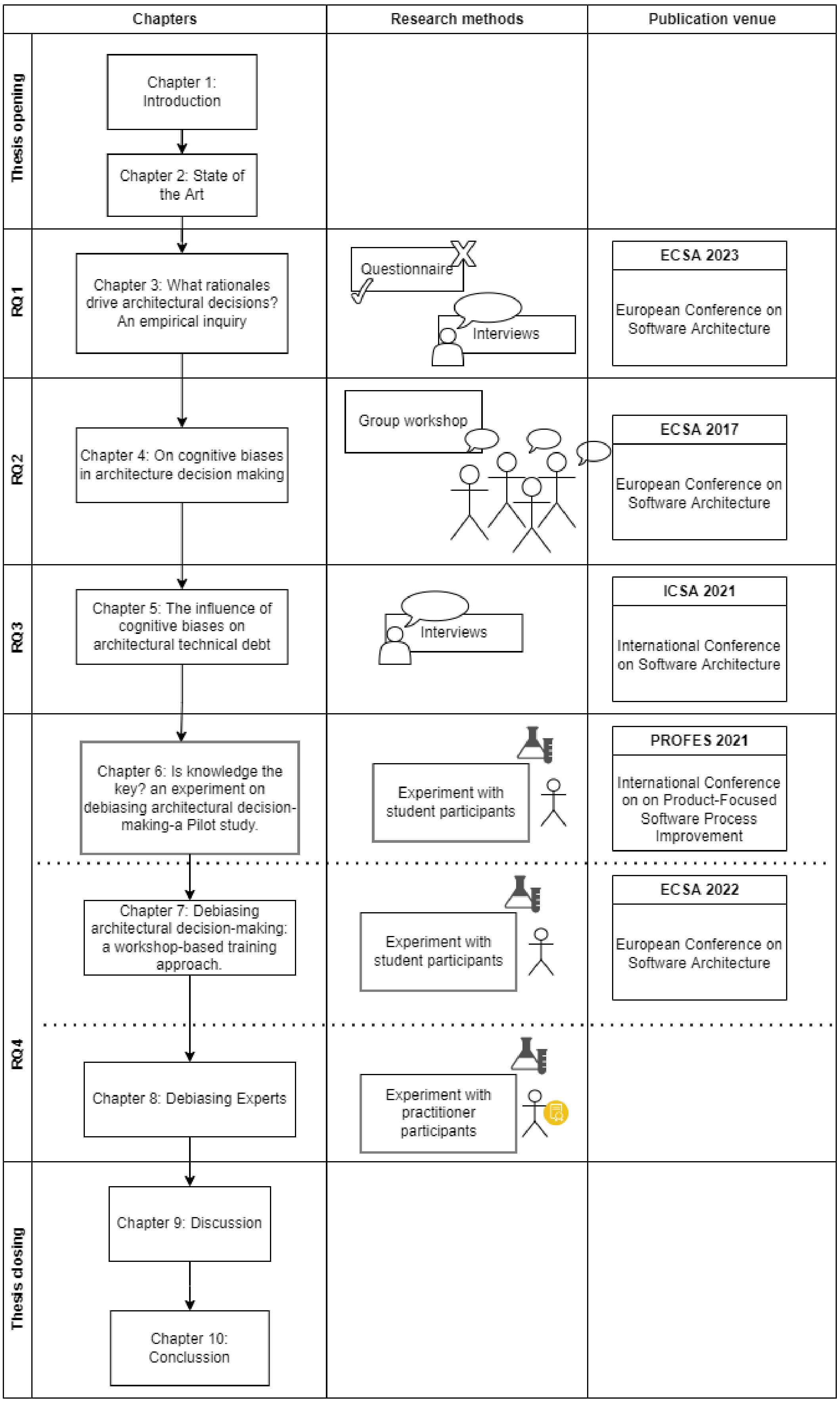
Chapters
Research methods
Publication venue
Thesis opening
Chapter 1: Introduction
Chapter 2: State of the Art
RQ 1
Chapter 3: What rationales drive architectural decisions? An empirical inquiry
Questionnaire
Interviews
ECSA 2023
European Conference on Software Architecture
RQ 2
Chapter 4: On cognitive biases in architecture decision making
Group workshop
ECSA 2017
European Conference on Software Architecture
RQ 3
Chapter 5: The influence of cognitive biases on architectural technical debt
Interviews
ICSA 2021
International Conference on Software Architecture
RQ 4
Chapter 6: Is knowledge the key? an experiment on debiasing architectural decision-making-a Pilot study.
Experiment with student participants
PROFES 2021
International Conference on on Product-Focused Software Process Improvement
Chapter 7: Debiasing architectural decision-making: a workshop-based training approach.
Experiment with student participants
ECSA 2022
European Conference on Software Architecture
Chapter 8: Debiasing Experts
Experiment with practitioner participants
Thesis closing
Chapter 9: Discussion
Chapter 10: Conclussion

Figure 1.1. Thesis overview

Chapter 3 contains the following peer-reviewed article:

***Klara Borowa**, Rafal Lewanczyk, Klaudia Stpiczyńska, Patryk Stradomski, and Andrzej Zalewski. "What rationales drive architectural decisions? An empirical inquiry." European Conference on Software Architecture. Cham: Springer Nature Switzerland, 2023.*

- **RQ2: How do cognitive biases impact architectural decision-making?** is answered in **Chapter 4**. This chapter presents the results of a workshop conducted with software practitioners, where they discussed cases when cognitive biases impacted architectural decision-making. The results of this workshop consisted of an ordered list of cognitive biases in accordance with their perceived impact on architectural decision-making.

  Chapter 4 contains the published peer-reviewed article:

  *Zalewski, Andrzej, **Klara Borowa**, and Andrzej Ratkowski. "On cognitive biases in architecture decision making." Software Architecture: 11th European Conference, ECSA 2017, Canterbury, UK, September 11-15, 2017, Proceedings 11. Springer International Publishing, 2017.*

- **RQ3: Does cognitive biases' impact on architectural decision-making cause architectural technical debt?** is answered in **Chapter 5**. This chapter contains information about an interview study conducted with software practitioners, which made it possible to understand which cognitive biases are likely to impact architectural technical debt, as well as the biases antecedents and this technical debt's consequences.

  Chapter 5 contains the following peer-reviewed paper:

  ***Borowa, Klara**, Andrzej Zalewski, and Szymon Kijas. "The influence of cognitive biases on architectural technical debt." 2021 IEEE 18th International Conference on Software Architecture (ICSA). IEEE, 2021.*

- **RQ4: How can the negative impact of cognitive biases on architectural decision-making be alleviated?** is answered in **Chapters 6, 7 and 8**. These three chapters showcase the development and validation of a debiasing intervention.

  Chapter 6 presents a pilot study with two groups of students - one of which took part in a simple debiasing presentation. This intervention turned out to be ineffective. However, analyzing the student's behavior resulted in a deepened understanding of cognitive bias influence, resulting in the proposal of a new set of debiasing techniques.

  Chapter 6 contains the following published peer-reviewed paper:

***Borowa, Klara**, Robert Dwornik, and Andrzej Zalewski. "Is knowledge the key? an experiment on debiasing architectural decision-making-a Pilot study." Product-Focused Software Process Improvement: 22nd International Conference, PROFES 2021, Turin, Italy, November 26, 2021, Proceedings 22. Springer International Publishing, 2021.*

Chapter 7 showcases the results of a debiasing workshop designed based on the findings from Chapter 6. 12 groups of computer science students took part in this experiment. The student groups were given two tasks to design an architecture - before and after participating in a debiasing workshop. The results were positive overall in most groups, which validated the workshop as useful for teaching students.
Chapter 7 contains the following published article:

***Klara Borowa**, Maria Jarek, Gabriela Mystkowska, Weronika Paszko, Andrzej Zalewski "Debiasing architectural decision-making: a workshop-based training approach." European Conference on Software Architecture. Cham: Springer International Publishing, 2022.*

Chapter 8 contains a **yet unpublished** experiment with expert practitioners. 18 practitioners from 3 different countries (Poland, Germany, and Brazil) took part in this experiment. Practitioners were divided into a control and experimental group, with the experimental group participating in the debiasing workshop. Overall, the impact of biases was smaller for the experimental group participants.

- In **Chapter 9** the overall results and their implications are discussed.
- **Chapter 10** contains the thesis conclusion.

2. State of the Art

The purpose of this chapter is to present research related to this thesis. It is worth noting that while starting this research, the author conducted a systematic literature review on the topic of cognitive biases in software engineering. This systematic review is detailed in the author's Master's Thesis [30]. This chapter contains both information present in that thesis as well as data on new relevant literature that was published in the following years.

Section 2.1 explains the concept of cognitive biases and the dual process theory. Section 2.2 showcases previous research on cognitive biases in the software engineering domain. Then, Section 2.3 explains the current state of research on cognitive biases in architectural decision-making. Finally, Section 2.4 contains information about debiasing, i.e., mitigating the impact of cognitive biases.

2.1. Cognitive biases

William James is considered a precursor of the **dual process theory** in psychology due to his work "The Principles of Psychology," where he defines various ways of human reasoning, two of them in particular: associative and rational [31].

Association in the human mind, according to James, is simply a perceived (not always factual) connection between certain things in the mind. It is described as heavily impacted by an individual's past experiences, e.g., "*that twenty experiences make us recall a thing better than one*" [31].

James explains that for reasoning to be rational, two steps must be taken: a specific piece of information must be perceived, and the consequences of this information must be deduced [31].

In his book, James gives the following example: "*Suppose I say, when offered a piece of cloth, " I won't buy that; it looks as if it would fade," meaning merely that something about it suggests the idea of fading to my mind, – my judgment, though possibly correct, is not reasoned, but purely empirical; but, if I can say that into the color there enters a certain dye which I know to be chemically unstable and that therefore the color will fade, my judgment is reasoned. The notion of the dye which is one of the parts of the cloth, is the connecting link between the latter and the notion of fading.*" [31].

In this thesis, this combination of "information" and "deduction" is considered necessary for a logical, not biased, rationale. **While "rationale" has the broad meaning of "reason for**

decisions," only rationales resulting in a logical deduction based on a fact can be considered rational, i.e., non-biased.

Posner and Snyder were the first researchers to distinguish between automatic and conscious thinking [32]. According to them [32], the automatic processes in the human mind occur when:

- the individual has no intention to consciously think about something,
- when there is no interference from other mental activities.

Posner and Snyder also note that the reason for such automatic thinking is the limited capacity of the human mind for conscious thought [32].

These concepts were developed further by Tversky and Kahneman, who introduced the term cognitive biases, which are systematic errors resulting from the heuristics of the human mind, used to simplify judgment operations [33]. To explain the source of cognitive biases, Kahneman's take on the dual process theory must be explained. He proposed a model of the human mind as one consisting of two systems: Systems 1 and 2. System 1 is fast - it operates on associations and various other heuristics. System 2, however, is capable of rule-based logical reasoning, yet slow. Cognitive biases occur when judgments are made solely based on System 1 [12].

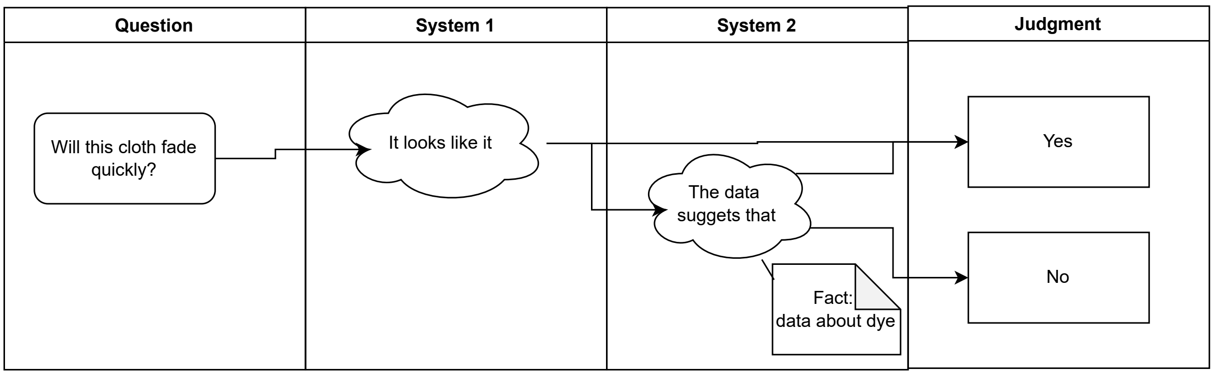

Figure 2.1. Dual-process judgment

The idea of two Systems fits with James' views on rational judgment. Figure 2.1 illustrates how James' example of judgment of a cloth's color fading could be explained by Khanemen's dual process theory [12]. In the case of biased judgment, System 1 would simply inform the individual that "it looks like it." For the judgment to be rational, System 2 would have to step in and asses the data about the dye used in the fabric. A significant change from James' original understanding is that, according to Kahneman, it is impossible to skip the "it looks like it" thinking provided by System 1. System 1 thinking is automatic, and as such, it always occurs. Instead, System 2 has to validate and correct these initial judgments. This concept is crucial for understanding how "rational decisions" are defined. **Rational decisions are ones made based**

on the judgment validated and corrected by System 2. This judgment must be based on a logical non-biased rationale (a fact and its logical consequence).

In the initial work of Tversky and Kahneman [33], three significant cognitive biases are introduced:

1. Representativeness – a cognitive bias that occurs because people unconsciously expect a sample to represent a different sample that they personally experienced. [34]. For example, a person in whose family most marriages ended in divorce may assume that divorce is more common in the general population than it is.
2. Availability – a heuristic that makes individuals make assumptions based on information most easily available to them [35]. For example, after repeatedly seeing an advertisement for a certain product, a customer is more likely to recall its name and thus may buy it.
3. Anchoring – is an individual's tendency to disproportionately consider only one possible value in making a decision. In particular – usually, the initial value was either presented to them, or they came up with themselves [36].

Since then, many more cognitive biases have been explored in a wide array of domains, e.g., tourism management [37], medicine [38] and software engineering [14].

To understand various biases' impacts on software engineering described later in this Chapter, it is necessary to know the definition of the following cognitive biases that have not been explained yet:

- Confirmation bias – the tendency to seek information that could possibly confirm one's belief while unintentionally avoiding information that may force a belief change [39]. For example, researchers may unintentionally seek only data sources that would confirm their research hypothesis while omitting ones that have a higher probability of disconfirming it.
- Over-confidence – The tendency to overestimate one's knowledge, skill, and accuracy, which makes individuals overestimate their control over one's environment [14]. For example, a person with a family member sick due to a specific illness, may perceive themselves as more knowledgeable on this illness than medical practitioners.
- Positive test bias – the tendency to set a hypothesis based on data the one knows will favor the hypothesis [40]. For example, a software tester that knows the test is positive for all even numbers between 2 and 100 may repeat this test for even numbers between 50 and 100.
- (Over) Optimism bias – the tendency to overestimate the probability of positive out-

comes [41]. For example, a developer may assume, without checking, that an API that he has to use will work exactly as he expects it to.

- Planning fallacy – the tendency to underestimate completion times of one's planned tasks [42].
- Egocentric bias – the tendency to overestimate one's own values, opinions, and contributions over the ones of other people [43].
- Fixation – while the term originated from Freud's theories on sexuality [44], fixation is now defined as overly focus on certain items, practices, and obstacles (even those that do not actually exist) [45].
- Framing effect – the tendency to interpret information depending on how it was presented [46]. For example, patients may choose a particular brand of medicine depending on how it is advertised or labeled.
- Halo effect – the tendency to evaluate unknown values of a certain item based on a small set of known values (usually the first ones noticed) [47]. For example, human beings often judge a person based on the first impression of their external appearance.

It is important to note that many of the definitions of biases were developed in parallel by researchers from various domains. As a result, the definitions of many cognitive biases, partially or even completely, overlap with each other. For example, positive test bias is a specific case of confirmation bias that is connected with performing formalized testing.

2.2. Cognitive biases in software engineering

Research on cognitive biases in various software engineering sub-domains is rather scarce. Mohanani et al., in their systematic mapping, which encompasses data up to 2016, found 65 papers relevant to the topic [14]. For years 2017-2018 an update of this list was performed as part of this thesis author's Master's thesis [30] - only three new relevant studies were identified this way. Since then, various new papers on the topic have been published, but the evidence from the mapping of Mohanani et al. [14] and its update [30] suggests that cognitive biases in software engineering are a topic with less than 10 papers published yearly.

This Section is divided into Subsections that focus on software engineering sub-domains that have been deeply researched in the context of cognitive biases:

- Requirements engineering (Section 2.2.1)
- Implementation (Section 2.2.2)

- Testing (Section 2.2.3)
- Software effort estimation (Section 2.2.4)

Other, less thoroughly researched topics are shortly explained in Section 2.2.5.

2.2.1. Cognitive biases in requirements engineering

Brownie and Ramesh, in their paper explaining phases of requirements determination, are the first to account for the possible impact of cognitive biases in the requirements domain [48]. In their paper, they explore how the three heuristics originally described by Tversky and Kahneman [10] may impact an analyst. They provide examples for all of them, e.g., for anchoring: "An analyst designing a user interface relies on past interfaces he has designed, fails to adjust adequately to current user needs, and delivers an interface that is not functional" [48].

Pitts and Browne explain various ways in which cognitive biases are likely to impact an analyst in the process of requirements engineering [49]:

- Anchoring and overconfidence may make them less likely to explore alternatives.
- Availability may limit exploration for new requirements to the requirements based on fresh memories from recent projects.
- Representativeness can make analyst see similarities with their previous project despite no evidence supporting this similarity.

Chakraborty et al. [50] present a model of requirements elicitation and suggest that the cognitive biases researched by Brownie and Ramesh [48] have an impact on the "Mental Model Creation" phase. One consequence of such impact may be overconfidence, making the users overestimate their business knowledge [50].

The author of this thesis has co-authored a peer-reviewed paper on the biases' impact on requirements elicitation. In it, based on an experiment with student participants, core biases that require further research in this domain have been pinpointed [51]. This filled a significant research gap, since previous research did not empirically assess which cognitive biases should be researched in this domain. Previously, Brownie and Ramesh [48], Chakraborty et al. [50], and Pitts and Browne [49] researched only the three cognitive biases defined originally by Tversky and Kahneman [10].

2.2.2. Cognitive biases in implementation

The first research paper that noted the possibility of cognitive biases' impact on any aspect of software development, was written by Stacy and Macmillan [52] and focused on software developers' everyday programming tasks. In particular, they discussed the following three biases and their impact [52]:

- Representativeness may make developers' judgment on the probability of certain events imprecise. For example, they may wrongly consider one output of independent random values to be more probable than another based on the output's contents.
- Availability may make developers use code constructs (e.g., long variable names) solely based on the fact that they either recently saw them in a project's code, or because they often see them.
- Confirmation bias may make finding errors in the code harder - developers may spend a large amounts of time searching for errors in the part of the code that are irrelevant to the error, since confirmation bias made them believe that this particular fragment of the code is faulty.

Another aspect of implementation that can be impacted by cognitive biases is artifact reuse. Parsons and Saunders [53], as well as Allen and Parsons [54], have performed empirical experiments which proved that developers are likely to anchor on reusing preexisting code. This may lead to the spread of existing errors, although this effect seems to affect novices more than experienced practitioners [53].

Chattopadhyay et al. [20] performed a field study with 10 developers, whom researchers observed during their everyday work. They found that about 70% of actions with a negative outcome could be associated with a cognitive bias. This research resulted in various notable findings - such as which biases impact developers during implementation and their consequences.

2.2.3. Cognitive biases in testing

Leventhal et al. [40] were the first to explore how cognitive biases may impact testing. They performed a set of experiments to asses whether students and professional testers are impacted by positive test bias while performing testing tasks. In all groups, positive test bias was present. Although it was the strongest in the case of junior students (in comparison to advanced students and practitioners).

Subsequently, work on exploring the impact of biases on testing was largely expanded by Calikli and Bener [55] [56] [57] [58]. In their various empirical evaluations [55] [57] [58], they proved the impact of confirmation bias during testing activities on both student [55] and practitioner [58] participants. This effect can be summarized as the following: while testing's goal is to find various errors, humans are likely to unconsciously seek to prove the software's correctness. As such, they are likely to perform unnecessary tests that they expect to return no information about any errors. As such, a confirmation bias' consequence is that the code becomes more defect prone [58]. Finally, Calikli and Bener [56] developed a metric scheme for measuring confirmation bias' levels and found a correlation between it and software defect rate.

Further exploration of confirmation bias' impact was performed by Salaman et al. [18] who, through a grounded theory study, developed a list of 20 antecedents of confirmatory and dis-confirmatory testing. The disconfirmatory antecedent that study participants mostly mentioned was "project experience". Previous experiences in the project allowed practitioners to recognize an increased number of edge cases that are more likely to result in failed tests.

2.2.4. Cognitive biases in software effort estimation

Jørgensen and Sjøberg [59] were the first to note the possibility that three cognitive biases described initially by Tversky and Kahneman [33] may impact software effort estimation. Moløkken and Jørgensen [60] focused on countering the effect of optimism bias that caused overly small estimates for project tasks. They discovered that group discussion on estimates, instead of combining individual estimates, is effective in countering over-optimistic predictions Moløkken and Jørgensen [60] [61] [62].

Additional research from this group of researchers has led to many findings, such as:

- Experts with technical knowledge are more likely to provide over-optimistic estimates for two reasons: (1) They analyze the problem from an "insider" perspective and, as such, are less likely to look at historical data; (2) They are perceived as more competent if they provide smaller estimates [63].
- In software projects, effort estimates are often backed by over-confidence bias. Practitioners may be over-confident for such reasons as (1) lack of feedback; (2) having solely an "inside" perspective; (3) trying to maintain an image as a more skilled expert; (4) the project managers seemed to reward over-confident team members [64].
- Over-optimism bias may be the source of the "winner's curse", i.e. the situation when software companies, in an attempt to win a bid to get a client, underestimate the project's workload so much that they face problems in delivering the software [65].
- Psychology tests meant to measure the subject's level of optimism (ASQ and LOT-R tests) are a poor predictor of over-optimistic predictions in a software project [66].
- Paradoxically, work on risk identification increases levels of over-optimism and over-confidence

– since having more information about risks makes practitioners feel more in control of an unpredictable situation [67].

Work on improving the accuracy of effort estimates to mitigate over-optimism and over-confidence has additionally been expanded by other researchers. Shmueli et al. [68] found that predictions are improved by providing reference information about past projects to provide an "outside" perspective. Sheppherd et al. [29] found that intentionally anchoring practitioners on an initial overly-pessimistic estimate can be used as a tool to rationalize effort estimations as well.

2.2.5. Other impacts of cognitive biases on software engineering activities

Agile software development has been researched in the following contexts:

- In pair programming tasks, novice developers are more often anchored on the initial solution, since they are less likely to think of multiple possible solutions [69].
- Student developers that are part of agile teams, due to the egocentric bias, overestimate the effort they put into the project in comparison to other team members. This effect slowly disappears as they gain more experience working together [70].
- Fixation makes agile team members over-focus on certain agile practices (e.g. daily meetings) while losing focus on key agile principles (e.g. self-organising teams) [17].

Designer creativity has been shown to be impeded by the framing effect, which may make designers fixate on a given set of requirements. To counter this effect, it is possible to simply re-frame designers' thinking by formulating requirements in a "softer" way, which allows for creative freedom [15].

User experience has been theorized to be impacted by cognitive biases, like the halo effect which makes the user's first impression extremely important [71].

2.3. Cognitive biases and architectural decision-making

Software architecture may be defined in various ways [72], these include:

- Perry and Wolf [5] describe software architecture as a set of elements: (1) processing elements; (2) data elements, and (3) connecting elements.
- Bass et al. state that software architecture is "a set of structures needed to reason about the system" [73].
- Jansen and Bosch [6] describe software architecture as a set of design decisions.

The particular definition of software architecture "as a set of architectural design decisions" [6] is the one assumed in this thesis. Jansen and Bosch [6] define a single architectural design decision as a decision containing the following:

- Rationale – the reasons behind the decision.
- Design rules – which describe how further decisions should be made.
- Design constraints – describing what behaviors are prohibited.
- Additional requirements – requirements that new architectural decisions must fulfill.

The viewpoint on software architecture as a set of design decisions empowered software architecture researchers to focus on architectural decision-making. This led to the creation of decision-making guidelines [74], as well as intensive empirical research on architectural decision-making [21].

Human factors in architectural decision-making, in particular, have been thoroughly explored [75]. This included the observation that software architects may make decisions impacted by cognitive biases, first made by Tang [76]. Later, van Vliet and Tang [77] expanded this idea by showcasing how anchoring, the framing effect, and confirmation bias may impact software architecture. Additionally, Manjunath et al. [23] conceptualized at which phases of the decision-making cycle cognitive biases may happen. The research presented in this Thesis further expanded the knowledge in this domain.

2.4. Debiasing

Debiasing is the act of countering cognitive biases. As described by by Fischhoff [78], there are four levels of debiasing treatments:

- A: Informing about cognitive biases. In this case, the intervention consists simply of one-way communication with general knowledge about cognitive biases, e.g., a lecture.
- B: Informing about cognitive biases, focusing on which biases may affect the intervention participants, e.g. a lecture where software architects would be informed about the current research on cognitive biases in architectural decision-making.
- C: Informing about biases combined with personalized feedback, e.g., a lecture about biases in architectural decision-making and a practical workshop during which participants could train in decision-making techniques.
- D: Extensive long-term training designed to counter biases, e.g., after performing a C-level debiasing intervention, periodic training workshops could be performed.

In the realm of software engineering, research on debiasing techniques is scarce [14], although the need for such research was noted over a decade ago by Ralph [45]. One such successful technique is having analysts use procedural prompts for requirements elicitation [49]. Another beneficial debiasing intervention was performed by Shepperd et al. [29], which used the framing effect to anchor developers on pessimistic time estimates - which resulted in countering over-optimistic time predictions.

While there is previous research on improving design rationale in software architecture [79] [80], the work presented in this thesis is the first that introduced an intervention that was empirically proven to counter cognitive biases in architectural decision-making.

3. What rationales drive architectural decisions? An empirical inquiry

Article title	What rationales drive architectural decisions? An empirical inquiry
Authors	**Klara Borowa**, Rafal Lewanczyk, Klaudia Stpiczyńska, Patryk Stradomski, and Andrzej Zalewski
Publishing venue	European Conference on Software Architecture
Year	2023
MNiSW points	140
Contribution	Original research idea, research method design, questionnaire analysis, most of the paper writing

3.1. Preface

Although this paper was created as the last of all published papers presented in this thesis, it is presented first for a specific reason. This results from a crucial research gap that existed before its publication: there was no existing empirical research exploring what rationales motivate practitioners during architectural decision-making. Some of these rationales were expected to be connected with cognitive biases, possibly being cognitive bias antecedents. This, however, had to be proven.

This research was performed in two steps. The first one was gathering data through a questionnaire where practitioners had to answer two open-ended questions: (1) "What do you consider when making architectural decisions?" (2) "What do your colleagues consider when they make architectural decisions?". Participants were supposed to give a maximum of three rationales as answers to both questions. We gathered 63 answers by distributing these questionnaires.

This questionnaire's results were grouped into rationale categories, from which five major rationale groups emerged: Ease of use for development activities, Maintainability, Performance, Prior knowledge/experience in using the solution, and Time/deadlines. Additionally, we observed two phenomena: (1) the rationales differed depending on the practitioner's experience level, and (2) a significant amount of practitioners believe that their colleagues make decisions based on the same rationales.

To explore what are the reasons behind these rationales, we performed a series of 13 follow-up interviews with questionnaire participants who agreed to it. As a result, we found a set of "reasons behind the rationales," the three most prominent ones being: (1) Practitioner's prior experience, (2) Client focus, (3) Making one's life "easy," i.e., decreasing their workload.

These findings, when viewed in the context of possible cognitive bias impact, allow us to draw a broader picture of architectural decision-making. Clearly, practitioners mainly based their rationales on their own personal experiences - such a way of thinking is the basis for all three classic cognitive biases introduced by Tversky and Kahneman [33]. Representativeness bias makes one believe that their personal experience represents a norm (e.g., if a solution was appropriate once, it will always be), Availability makes them base assumptions on easily available information (namely, their own experience), and anchoring makes them unlikely to consider other points of view. Since the rationales motivating practitioners are possible cognitive bias antecedents, **the exploration of cognitive biases' effect on architectural decision-making is a natural extension of this research.**

3.2. Abstract

Architectural decision-making is a crucial concern for researchers and practitioners alike. There is a rationale behind every architectural decision that motivates an architect to choose one architectural solution out of a set of options. This study aims to identify which categories of rationale most frequently impact architectural decisions and investigates why these are important to practitioners. Our research comprises two steps of empirical inquiry: a questionnaire (63 participants) and 13 interviews. As a result, we obtained a set of rationales that motivated architects' decisions in practice. Out of them, we extracted a list of software quality attributes that practitioners were the most concerned about. We found that, overall, architects prefer to choose solutions which are familiar to them or that guarantee fast software implementation. Mid-career architects (5 to 15 years of experience) are more open to new solutions than senior and junior practitioners. Additionally, we found that most practitioners are not concerned about the quality attributes of compatibility and portability due to modern software development practices, such as the prevalence of using specific standards and virtualisation/containerization.

3.3. Introduction

Understanding software architecture as a set of architectural decisions (ADs) [6] draws our attention to the motivation underlying these decisions and - this way - the entire architecture. Design rationale, which is a component of ADs [81], consists of the knowledge and reasoning justifying design decisions[82].

The research on factors (including rationales) [83] that shape architectural decisions in practice is rather scarce and seems still far from being mature. The most recent papers by Weinreich et al. [84], Miesbauer et al. [85] and Tang et al. [82] that explore the motivations underlying practitioners' ADs are at least eight years old. These works are continued in more recent studies that investigate what software quality attributes (QAs) are discussed when choosing architectural patterns [86] and what technology features drive technology design decisions [87].

As the software development landscape changes rapidly, the general purpose of this study is to discover what rationales, and why, currently drive ADs in practice. Such results importantly extend our knowledge and understanding of architectural decision-making (ADM) by allowing researchers to focus their efforts on improving ADM on the basis of current needs and practices of architects. Additionally, we put an emphasis on QAs since they are a rationale subset that has been of major interest for researchers [88] [86] [89].

Such an aim is expressed by the following research questions:

- RQ1: What rationales most frequently influence architectural decisions?
- RQ2: Which software quality attributes are usually prioritised during architectural decision-making?
- RQ3: Why do practitioners prioritise these rationales?

In order to investigate the above problems we performed a two-phase inquiry. Firstly, we gathered data through a questionnaire. We obtained answers from 63 practitioners. Then, we presented the questionnaire's results to 13 practitioners during interviews. As a result of the questionnaire, we created a list of rationales (including quality attributes as given in ISO 25010 [90]) that practitioners of various experience levels (beginners, mid-career and experts) consider essential. As a result of the interviews, we found out that, depending on experience level, practitioners tend to prioritise different architectural options.

The rest of the paper has been organised as follows: Section 3.4 presents related work, Section 3.5 contains details about our research process and Section 3.6 the study's results. We discuss our findings in Section 3.7, present the threats to validity in Section 3.8 and conclude in Section 3.9.

3.4. Related Work

The notion that software architecture is a set of design decisions [6] has heavily impacted the field of software architecture [88]. To enable better decision-making, researchers have explored such areas as: human factors in ADM [83], AD models [81], mining AK [89], curating AK [91], tools supporting decision-making [92], techniques that can aid designers in the decision-making process [93] [94] and ADM rationale [82].

Numerous aspects make ADM an extremely challenging process. The traditional decision-making process, which includes listing all possible alternatives and their attributes, is impractical for software design decisions [95] because of the number of possible architectural solutions. Furthermore, practitioners can be overwhelmed by the time and effort required to find architectural information [96]. Additionally, an entirely rational design-making process is impossible as long as it depends on human beings, that are impacted by various human factors [83].

While there exist general guidelines [94] and various tools [88] for ADM, empirical research on ADM factors is scarce [83]. On the topic of the practitioners' rationale behind design decisions, several studies must be acknowledged. Firstly, the study of Tang et al. [82], reporting the results of a survey on practitioners' approach to architectural rationale. Researchers had practitioners choose the importance of generic rationales and optionally allowed participants to provide their own rationales. As a result, a list of 12 rationales indicated by practitioners was made. This study's results were later expanded by Miesbauer et al. [85] and Weinreich et al. [84] who performed interview-based studies through which the list was expanded to include 18 rationales in total. Soliman et al. [87] researched what technology features impacted technology design decisions. Bi et al. [86] took a different approach and researched which ISO 25010 software quality attributes [90] were most often discussed in the context of architectural patterns on the StackOverflow platform.

We found no recent empirical research focusing widely on ADM rationale more recent than eight years ago. As software technology evolves rapidly, the rationales could also change. Additionally, we found no studies on how rationales depend on architects' professional experience, which we believe could be relevant since junior and senior architects find different aspects of ADM challenging [97].

3.5. Method

Our research comprises two phases: questionnaire and interviews. The purpose of the questionnaire was to gather a larger sample of data that would enable us to answer RQ1 and RQ2. The interviews let us delve deeper into the meaning and implications of the questionnaire's results (RQ3). Another reason for using two data-gathering methods was to achieve so-called 'methodological triangulation' [98], which helps to strengthen the validity of our findings. The overview of the study process is presented in Figure 3.1. The questionnaire questions, a summary of questionnaire results, the interview plan, and interview coding details are available online [99].

3.5.1. Questionnaire: data-gathering

The questionnaire's [99] design was simplistic in order to avoid discouraging practitioners from taking part and to avoid biasing the results by suggesting any specific answers. The questionnaire was divided into four main sections:

1. Participant data: age, gender, education, years of experience in software development, role in the company, company size, company domain.
2. An open-ended question to provide a maximum of three most often used rationales for architectural decisions, according to the participant's personal experience.

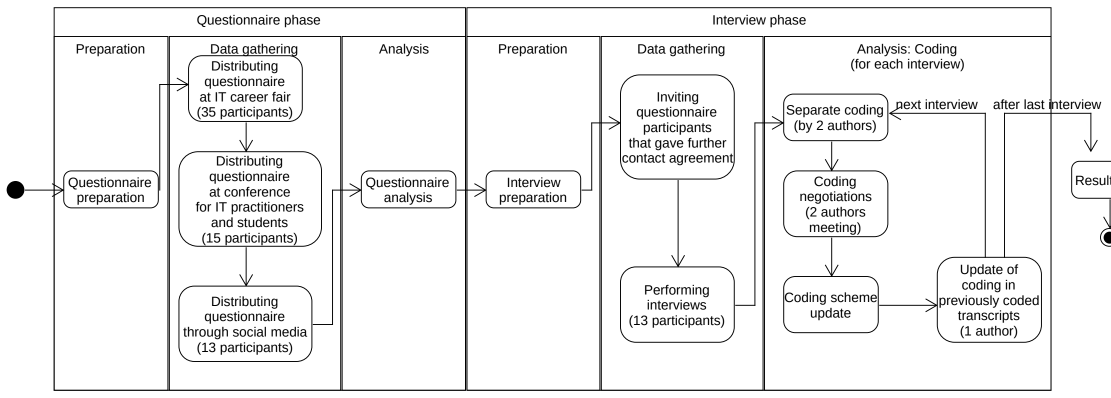
Questionnaire phase
Interview phase
Preparation
Data gathering
Analysis
Preparation
Data gathering
Analysis: Coding (for each interview)
Questionnaire preparation
Distributing questionnaire at IT career fair (35 participants)
Distributing questionnaire at conference for IT practitioners and students (15 participants)
Distributing questionnaire through social media (13 participants)
Questionnaire analysis
Interview preparation
Inviting questionnaire participants that gave further contact agreement
Performing interviews (13 participants)
Separate coding (by 2 authors)
next interview
after last interview
Coding negotiations (2 authors meeting)
Coding scheme update
Update of coding in previously coded transcripts (1 author)
Results analysis

Figure 3.1. Study phases

3. An open-ended question to provide a maximum of three most often used rationales for architectural decisions by the participant's colleagues. We asked this question to investigate if the participants believed that other practitioners have different priorities from them.
4. An optional section containing the option to provide an email and give consent for further contact from the researchers.

In order to obtain samples for the study, we distributed the questionnaires in three different locations:

1. During a 3-day long IT career fair at our faculty, where representatives of over 50 companies were present. We approached each stall and gave a physical copy of the questionnaire to the practitioners that were advertising their companies. We obtained 35 completed questionnaires at this event.
2. During an IT conference for practitioners and students, where representatives from over 60 companies were present. We used the same strategy as the one during the career fair and obtained 15 additional completed questionnaires.
3. We made the questionnaire available online and posted it on our personal social media accounts; this led to additional information from 13 participants.

In total, we obtained data from 63 participants. A summary of the participants' demographic data is presented in Figure 3.2, and their employers' companies' domain and size in Figure 3.3.

3.5.2. Questionnaire: analysis

To analyse the questionnaire, we performed the following actions:

1. We divided the participants into the following groups: beginners (under five years of

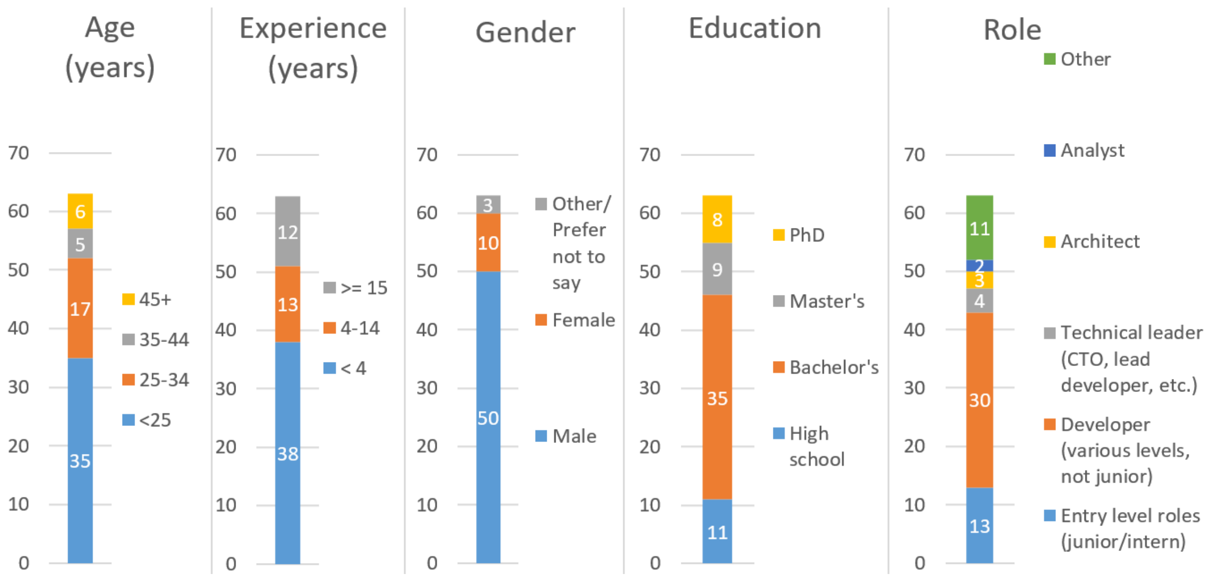

Figure 3.2. Questionnaire participants

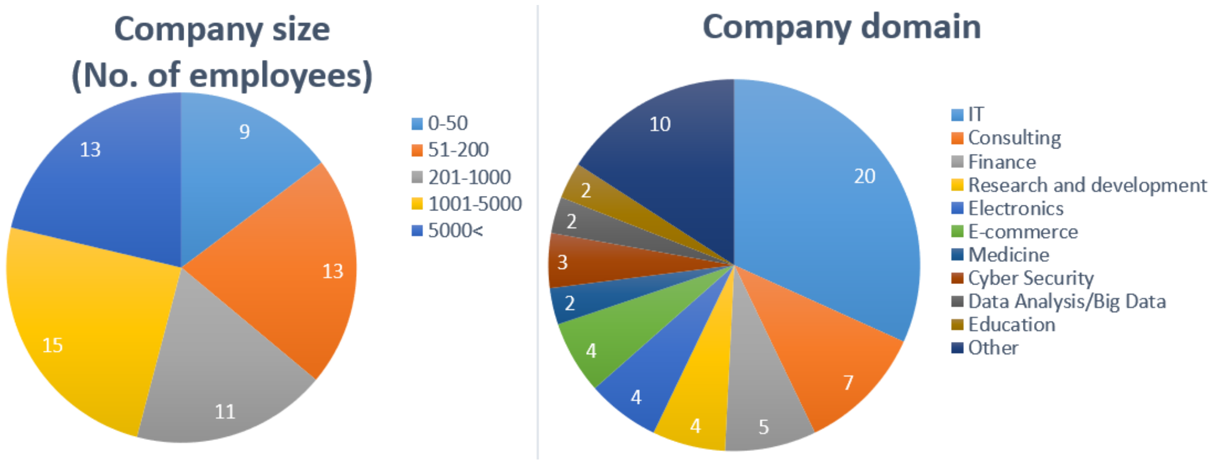

Figure 3.3. Questionnaire participants companies

experience), mid-career (5 to 14 years of experience), and experienced (15 or more years of experience) practitioners.

2. We extracted the answers about the participants' as well as their colleagues' rationales and analysed them separately.
3. For each of the six combinations of the above groups (participants' experience level and their own/colleagues' rationales) separately, we classified the rationales (even if they were worded differently) into categories. When applicable, we used the ISO/IEC 25010 [90] software quality attributes as the rationale categories. We grouped rationales into categories, since participants often used different words to explain the same factors influencing their decision-making. A rationale category groups rationales that are similar to such a degree that we found them almost indistinguishable. For example, we categorised all of the following

Table 3.1. Interview participants

No.	Gender	Age (years)	Experience (years)	Education	Role	Company size (employees)	Company domain
1	Male	23	1	Bachelor's	Software Engineer	1001-5000	Infrastructure monitoring
2	Male	22	1	Bachelor's	C++ Developer	51-200	Power Engineering
3	Male	45	22	PhD	Company owner	0-50	IT, Data Science
4	Male	23	1	Bachelor's	Python Backend Developer	51-200	Software House
5	Male	22	1	High School	Junior Developer	1001-5000	E-commerce
6	Male	23	3	Bachelor's	Junior Java Developer	over 5000	Consulting
7	Male	24	4	Bachelor's	Software Engineer	51-200	Finance
8	Male	31	5	Master's	Software Developer	1001-5000	Electronics
9	Male	45	20	PhD	Architect	over 5000	Commerce
10	Female	25	3	Master's	NLP Engineer	over 5000	R&D
11	Male	41	20	PhD	CTO	201-1000	Finance
12	Male	28	5	High School	Senior Testing Engineer	201-1000	Videogame development
13	Male	32	6	Master's	Senior Software Engineering Manager	over 5000	FMCG

as "Time/Deadlines": "time that we will waste on it; how much time there is to do it; time available to create the software; Number of hours required to write the functionality; time-consumption of making the solution; time-consuming; deadline to deliver the project; time available; time". When rationales were only related to each other, like for example "Documentation" and "Maintainability", we did not categorise them together. Table 3.3 summarises the questionnaire analysis results.

3.5.3. Interviews: data gathering

Based on questionnaire data analysis, when creating the interview plan [99], we focused on the following categories of observations:

1. The rationales common for 20% of the participants of each professional experience level.
2. Quality attributes of generally low interest to the architects, namely, attributes mentioned by fewer than 5% of all the participants.
3. Cases in which answers varied among architects of different experience levels. For example, some rationales were over the 20% cutoff score in one group but not in all of them.

We presented the results from the questionnaire in which the above cases occurred to the interviewees. Then, we asked them about the reasons behind the observed level of importance of these rationales for specific architects' experience groups.

All 13 interviewees were recruited from the questionnaire participants. We invited to a follow-up interview all participants that consented to a follow-up interview in the questionnaire. Table 3.1 presents the overview of the interviewees' characteristics.

3.5.4. Interviews: analysis

The interview recordings have been transcribed. Then we coded the transcripts by following the subsequent steps:

1. Two separate authors coded the same transcript using the descriptive coding method [100]. This means that segments of the transcripts, which contained a relevant piece of information, were labelled with a code that described its type of content. We started with an empty list of codes, to avoid biasing the results towards our own ideas, and allowed the codes to emerge during the coding process.
2. Both coding authors met to negotiate their coding [101] — they made changes to the coding until reaching a unanimous consensus.
3. An updated list of codes was created as a result of the coding meeting.
4. One of the authors re-coded previously coded transcripts with new codes if they emerged during the current analysis step.
5. The above steps were repeated for each interview transcript.

Codes are summarised in Table 3.2. After coding all transcripts, we analysed and discussed the coded segments to draw conclusions.

3.6. Results

Table 3.3 presents the questionnaire results. As explained in Section 3.5.3, we consider the rationale as important to a given group of architects if it was indicated but at least 20% of them. Additionally, we focused on software quality attributes that were mentioned by less than 5% of the participants and the variation in rationale prioritisation in different groups of participants.

3.6.1. RQ1 & RQ2: Most frequent rationales and prioritised software quality attributes

The rationales that most frequently occurred in the questionnaires (over 20% of participants) were:

Table 3.2. Codes

Code	Description	Number of occurrences	Number of interviews where code occurred
EX	Perspective/performed tasks change with the developer's experience	58	13
CLNT	Recognising client's needs, focusing on the client's benefit.	31	12
EASY	Participant mentions how important ease of use for development/maintenance is in the project	28	13
FUT	Thinking about what effects the choice will have for the project	29	12
D	Focusing on the deadline/ how much time something will take	23	9
FAM	Choosing something based on one's familiarity with it	25	9
IMP	The rationale was omitted because it is 'obviously' important	18	9
EMP	Thinking how the choice will impact other people	15	10
CR	Focusing on personal growth	15	8
OUTDATED	The rationale does not require much thought because it is handled by newer technology	13	8
NEG	The participant disagrees with other practitioners' opinions (from the questionnaire)	11	7
EDU	Described behaviour is an effect of education	10	7
RARE	The participant considers something as niche or unimportant	9	5

Table 3.3. Questionnaire results. ISO/IEC 25010 quality attributes are marked by a **bold** font.

		Sum		Beginners		Mid-career		Experienced	
No.	Rationale category	Participants	Colleagues	Participants	Colleagues	Participants	Colleagues	Participants	Colleagues
1	Ease of use for development	23	11	16	7	2	0	5	4
2	**Maintainability**	15	2	12	1	2	1	1	0
3	**Performance**	14	6	13	6	0	0	1	0
4	Prior knowledge/experience	14	14	11	9	1	2	2	3
5	Time/deadline	12	8	10	6	1	0	1	2
6	**Reliability**	10	4	6	3	2	1	2	0
7	Development Project Environment	9	2	4	1	3	1	2	0
8	Cost	8	9	5	7	1	0	2	2
9	Popularity	8	8	7	5	0	1	1	2
10	Scalability	7	3	4	3	2	0	1	0
11	Business/customer requirements	7	5	4	4	1	0	2	1
12	Documentation	6	4	6	4	0	0	0	0
13	**Usability**	5	0	3	0	2	0	0	0
14	**Security**	5	2	3	2	2	0	0	0
15	Aesthetics/UX	5	2	1	1	2	0	2	1
16	Fit with existing systems/project	5	7	4	4	0	1	1	2
17	Decision-making methodology	5	4	0	0	2	1	3	3
18	Testability (simplicity of writing tests)	4	0	3	0	0	0	1	0
19	Level of complexity of the problem/system	4	1	4	1	0	0	0	0
20	Expertise of more experienced colleagues	4	1	4	1	0	0	0	0
21	**Functional Suitability**	3	1	2	1	0	0	1	0
22	Availability of packages	3	0	1	0	0	0	2	0
23	Team members' preferences	3	4	2	4	0	0	1	0
24	**Portability**	2	2	2	1	0	0	0	1
25	System life expectancy	2	0	0	0	0	0	2	0
26	I want to add new skill to my resume	2	1	2	1	0	0	0	0
27	**Compatibility**	1	2	0	0	0	0	1	2
28	Return on Investment (ROI)	1	1	1	1	0	0	0	0
29	Market expectations	1	0	1	0	0	0	0	0
30	Available human resources/money	0	4	0	2	0	2	0	0
31	Bus factor	0	1	0	1	0	0	0	0
32	"It works so I should use it"	0	1	0	1	0	0	0	0
33	My colleagues have the same rationales as me	0	19	0	13	0	4	0	2

1. **"Ease of use for development"** was the dominant rationale for almost all groups of participants. Over 40% of the beginner and expert groups believed that it was important. However, this was not the case for mid-career practitioners, where only 15% mentioned this rationale.
2. The quality attribute of **"Maintainability"** was the second most often indicated rationale, which was mentioned by 24% of the participants. This was due to the beginners' insistence that this rationale is important (30% of them), though it was not similarly prioritised by mid-career practitioners (15%) and experts (8%).
3. Both the quality attributes of **"Performance"** and **"Prior knowledge/experience"** were mentioned by the same number of practitioners overall (22%). **"Performance"**, similarly to **"Maintainability"**, was important to beginners (33%) but not to mid-career (0%) and to expert practitioners (8%). **"Prior knowledge/experience"** of the solution, in the same way as **"Ease of use for development"**, was prioritised by both beginners and experts (over 20% in both groups) but not by mid-career practitioners (only 8%).

Rationales that were overall mentioned by less than 20% of the participants but were important for a particular group of practitioners (over 20% of that group):

1. **"Time/deadline"** is a rationale that was mentioned by 26% of beginners but less often by mid-career and expert practitioners (8% in both groups).
2. **"Development Project Environment"**, which refers to various aspects of management and organisation of development project (e.g. company standards, client specifics) or current possibilities (available technologies), was important to mid-career practitioners (23%) but less so to beginners (10%) and experts (16%).
3. A **"decision-making methodology"** was by experts (25%) but only a few mid-career practitioners (8%) and no beginners.

Three software quality attributes were mentioned by less than 5% of the participants: **Compatibility** (1 participant), **Portability** (2 participants) and **Functional Stability** (3 participants).

Finally, when asked about their colleagues' rationales, most participants wrote unprompted in their questionnaires that **their colleagues are motivated by the same rationales as they are themselves** (30%). These were not cases of copying the same answers from one question to another but literally writing a statement about one's colleagues. This answer dominated the beginner (33%) and mid-career (31%) groups but occurred less frequently in the expert group (17%).

3.6.2. RQ3: Rationales' origins

By analysing the interviews, we found a key set of rationales' origins. Some rationales and rationale origins may slightly overlap (e.g. "Time/deadlines" rationale and "fear of deadlines" rationale origin). This was the case when participants listed both a rationale in the questionnaire and a rationale's origin in the interviews. The rationale's origins include (number of code occurrences overall/number of interviews where code occurred):

1. **Practitioner's experience(58/13)**: The primary origin of the practitioners' rationales were their previous experiences. Beginners had limited experience, and to avoid the risk of not performing their duties efficiently, they preferred the solutions which they had used previously - because of that, "Ease of use for development" and "Prior knowledge" turned out to be the prevailing rationale for them. As one of the participants stated: "(...) [junior developers] are such fresh people, it is certainly much more convenient. Because, well, since it's easy to learn something [how to use a solution], it's easy to reach the right level quite quickly."
   Experts with significant experience also prioritised these rationales but for different reasons – they already had knowledge that they were confident in, so they did not feel the need to try new solutions and leave their comfort zone, e.g. "Maybe more experienced people who worked a long time with a certain technology change it less often than people who are just entering the IT market (...), but feel comfortable with certain technologies and have been comfortable working with them for many years."
   The exception to this effect were mid-career practitioners who were most likely to possess the knowledge and willingness to discover new solutions. As a participant said: "Maybe the moderately experienced people are neither those very experienced people who have been working in a particular technology for a longer period of time, but those who change it more often and maybe they see that it is not that difficult, they are used to changing technologies."
   The practitioner's experience influence was also crucial for choosing the "Time/deadline" and "decision-making methodology" rationales. Beginners feared the possible consequences of missing a deadline more than other practitioners. Hence, they indicated the "Time/deadline" rationale more frequently than more experienced architects, e.g. "People with more experience are more assertive when it comes to deadlines and are able to say 'no' when they know that it is simply impossible to do something in a certain time, and those with less

experience may also not be so sure that this is the moment that it is worth saying ’no’ and not doing something, they are afraid of the deadline.”

However, using a “decision-making methodology” as their rationale’s foundation was only possible to experienced practitioners due to their greater knowledge, e.g. “ (...) we [the architects] are just getting used to such methodologies, acquiring them, so we will only use them after some time.”

2. **Client focus (31/12)**: Various rationales originated from the endeavour to meet the client’s needs. Practitioners often prioritised “Ease of use for development” and “Time/deadline” rationales because they strived to deliver new functionalities to the client as soon as possible, e.g. “(...) recently there has been a lot of emphasis on time to market and deadlines for implementing individual functionalities, which are usually short.”

   Similarly, the “Development Project Environment” had to be considered to satisfy the client’s needs. Even if two projects appeared to be the same, the environment often made a difference in its development. As one participant stated: “Otherwise, seemingly the project sounds the same, but in practice, the client often wants something completely different than the previous one.”

   Additionally, “Performance” was seen generally as a key software quality attribute from the client’s perspective, since weak software performance (software freezes, long waiting times, etc.) was seen as very problematic to the clients, e.g. “(...) usually the performance of the system is related to the comfort of use, so it seems to me that this is also the reason why performance is an important criterion.”

3. **Making one’s life “easy” (28/13)**: Generally, practitioners choose solutions that they believed would make their work as effortless as possible. This was not only related to the “Ease of use for development” rationale but also “Prior knowledge” (the source of information about what is “easy”) and “Maintainability” (minimisation of future work). As one participant stated: “Some developers are lazy, which means that solutions that are easier to maintain often scale easier and perhaps require less work or less mental effort to add a new feature or to fix a bug.”

4. **Thinking of the project’s future (29/12)**: In general, practitioners were aware of the software life-cycle and knew that “Maintainability” could impact the amount of effort they would have to put into maintaining the system in the future. However, “Ease of use for development” was also a rationale impacted by this factor. Practitioners believed that if it

is easy to use a given solution, it will also be easier to find, hire and train new employees that would work on the project in the future, e.g. "(...)the ease of training new employees to work, whenever the software is easier to develop and is based on popular technology or the code is transparent, it is easier to introduce someone new here."

5. **Fear of deadlines (23/9)**: The fear of missing a deadline had a major impact on beginner practitioners. This was not the case for mid-career and expert practitioners since they already had experiences with missed deadlines in their careers and had the capacity to imagine how such a situation could be handled. For example: "I think it's because the more experienced ones, I also know that this is how managers and programmers work, as well as project managers, that they know that this deadline is set with some reserve."
6. **Familiarity with a particular solution(25/9):** Prior experience with a particular solution was the main source of architectural knowledge. Since it is rarely possible to explore all the possible alternatives, prior experiences are the primary source of information, e.g. "Architecture, all engineering, in general, is based on experience, and experience means things that we brokne in previous designs, in previous products. And on this experience, which looks so negative, but is nevertheless building our knowledge, we base what we create in the future.".
7. **"Obviousness" (18/9)**: In the case of the "Functional Stability" quality attribute, some practitioners expressed the opinion that the importance of this rationale is simply obvious, and as such, there was no need to mention it in the questionnaire, e.g. "It's [Functional Stability] also so mundane and part of such day-to-day work that maybe we don't tie it to the architecture.".
8. **Empathy (15/10)**: "Ease of use for development" and "Maintainability" were often prioritised because of the practitioners' awareness that their colleagues will have to maintain and further expand a system in the future, e.g. "It should be done in such a way that I would not hurt myself or that it would not be painful for my colleagues to maintain. I see in this perhaps some form of empathy.".
9. **Personal growth (15/8)**: Mid-career practitioners did not prioritise "Ease of use for development" and "Prior knowledge" rationales, as beginners and experts did. Our participants pointed out that mid-career practitioners are in a specific professional situation where they can already feel confident in their basic knowledge (unlike beginners) but strive to learn about new solutions to further develop their careers (unlike experts). As one participant

stated: "(...) resume driven development, i.e. we choose those technologies that will look nice in the CV, or that will make us learn something.".

10. **New technology handles the problem (11/7)**: In the case of the "Compatibility", and "Portability' quality attributes, practitioners believed that new technologies already solved most problems related to these rationales. In the case of "Compatibility", currently, existing standards are widely used, and compatibility problems are rare. As a participant stated: "(...) because everything is somehow compatible with each other, only a matter of certain calling some services(...)".
    Similarly, the widespread use of virtualisation and containerisation solved most problems with "Portability", as a participant stated: "(...) because practically everything can be uploaded, containerized".
11. **Practitioner's education(10/7)**: "Performance" was stated to be a rationale prioritised by beginner practitioners that recently finished their degrees in a field related to Software Engineering. This was due to the focus on the use of optimal data structures and algorithms during their studies, e.g. "(...) during studies and in earlier educational programming, a lot of emphasis was placed on making these solutions work quickly. I even had one subject where we were judged on how many minutes it took to run a program, so it stuck in my head a bit.".
12. **Perception of the quality attribute as unimportant(9/5)**: Some participants stated that in the case of the projects that they worked on, "Compatibility" and "Portability" quality attributes were not important. For example, the project was targeted to work on a very specific platform, as the participant stated: "(...)projects are created, for specific hardware or for specific platforms, not multi-platform solutions."

3.7. Discussion

Two top rationales that were not quality attributes were "Ease of use for development" and "Prior knowledge / experience". This result is similar to the findings of Miesbauer et al. [85] and Weinreich et al. [84] who found that the most influential rationale was "Personal experience / Preferences". This implies that the current trend of researching human factors in ADM [83] [88] is appropriate for further understanding and improving ADM. To be more specific, it seems that practitioners prioritise minimising their own and their colleagues' workload, both in the short and the long term. This fits with the principle of "Simplicity – the art of maximising the amount of work not done" [102] from Agile software development. However, if done inappropriately, this can lead to consequences such as incurring architectural technical debt [103].

The quality attributes of "Maintainability" and "Performance" were perceived as the most important out of the set of ISO 25010 software quality attributes [90]. This matches the findings of Bi et al. [86] who found these to be the most often discussed quality attributes in the context of architectural patterns. We further explain this phenomenon since we found that beginner practitioners emphasise these rationales more than experts. In the case of "Maintainability", it seems that they wanted to avoid their own future workload, which may be perceived as an intimidating perspective. In the case of "Performance", beginners followed the knowledge acquired during their formal education and the emphasis of scholars on algorithmic efficiency.

Additionally, we found that practitioners in general do not put an emphasis on the quality attributes of "Portability" and "Compatibility". Modern technologies deliver solutions that well-address both these issues. In the case of "Portability", there are many efficient tools that resolve such problems: virtualisation, containerisation or frameworks for building multi-platform applications. Furthermore, in some fields (like developing console video games), the hardware on which the software will be run can be accurately predicted. Challenges with "Compatibility" have been overcome mostly through the standardisation of the technologies used by practitioners; for example, in the case of web applications, a REST API between the front-end and back-end layers is a predictable solution that most would choose by default.

Finally, we discovered that depending on experience level, practitioners have a significantly different mindset when it comes to ADM. Beginners are greatly influenced by a fear of the unknown: they fear that it would be too hard to develop the software, or to maintain it later, to learn new solutions during the projects, and the consequences of unmet deadlines. Experts experience less fear of deadlines but put an emphasis on ease of development to make their colleagues' work easier and feel comfortable with their current practices. They were also the only group to use any decision-making methodologies, which they found natural if they gained enough knowledge. Lastly, mid-career practitioners are the most open to learning about new solutions and attempting not to use ones that are not considered "easy", to create bespoke solutions that would fit their clients the best.

3.8. Threats to validity

In this Section we describe three main kinds of threats to validity [98]:

Construct Validity To find the participants' rationales for architectural decisions, the possible methods of enquiry are either methods based on self-reporting or observation of the participants' work. We have chosen self-reporting methods (questionnaires and interviews) since that enabled us to obtain data from a greater number of practitioners. However, it is still possible that the participants' actual rationale may differ from those that they reported. For example, they may be impacted by cognitive biases [104] that they are not aware of.

Internal Validity To maximise the internal validity of our findings, the coding of the transcripts was always done independently by two authors. Then, both discussed the coding until they unanimously agreed on all codes. This was done to minimise the impact of the researcher's bias on the findings. However, it is possible that factors that we did not consider could play a role in practitioners' approach to decision-making, such as their company's size or domain.

External Validity We used convenience sampling since it is an extreme challenge to obtain a random generalisable sample of software practitioners. However, we strived to overcome this by providing data source triangulation [98]: we searched for participants from three different sources (two in-person events and one on social media). This resulted in a varied group of participants. Though, worth noting is that the sample may be biased towards less experienced practitioners, due to the majority of participants having less than 4 years of professional experience. Additionally, since our results partially match results from previous studies [84] [85] [86], it seems that our sample was big enough to give us outcomes also noticeable to other researchers.

3.9. Conclusion

In this study, we performed a mixed methods two-step empirical inquiry into the practitioners' rationale behind their architectural decisions. The three main contributions of this study are as follows:

1. A list of the most impactful rationales that influence practitioners' architectural decision-making;
2. An exploration of these rationales' origin;
3. The finding of how a practitioner's experience has a significant impact on how they make architectural decisions.

Future research could employ different research techniques to further confirm or disconfirm our findings. A survey on a random generalisable sample would be beneficial, as well as observational studies on practitioners that would explore their decision-making in real-time. In accordance to our findings, since experience level seems to be a major factor shaping who architects make their decisions researchers should take it into account during future research on ADM.

Practitioners could benefit from our study by understanding better the way they and their colleagues develop software architectures. The observation on the influence of experience on ADM should also be reflected in shaping a team's structure, e.g. it would be prudent to focus on having mid-career (between 5 and 14 years of experience) practitioners in their teams when working on an innovative project.

4. On Cognitive Biases in Architecture Decision Making

Article title	On cognitive biases in architecture decision making.
Authors	Andrzej Zalewski, **Klara Borowa**, and Andrzej Ratkowski
Publishing venue	European Conference on Software Architecture
Year	2017
MNiSW points	140
Contribution	Data gathering, data analysis, writing the paper's first draft, and making changes to the final paper.

4.1. Preface

This article was the first published from all the papers presented in this thesis. The inspiration to do this research comes from Kahneman's classic book "Thinking, Fast and Slow" [11] and the few existing pieces of research on cognitive biases in architectural-decision making at that time [22] [77].

Having basic knowledge about cognitive biases, we designed a simple workshop to gather data from 14 practitioners about their experiences with possibly biased architectural decisions. The main steps of the workshop were explaining the theory behind cognitive biases, gathering a list of experiences in which participants believed a bias occurred (without naming the specific bias), and having the participants rate how often such occurrences happened.

During analysis, we matched the participants' stories with bias definitions and identified a list of cognitive biases that impacted architectural decision-making. Additionally, we ranked how often they occurred using the participants' ratings. Finally, we analyzed how these biases may interact with each other and various real-world software development factors.

So far, this work, in particular, has impacted software architecture research - to this day, it has been cited 31 times, according to Google Scholar. Its main value lies in giving researchers a specific, ranked list of biases that they should consider in the context of software architecture. Previous work provided examples of biases impacting architectural decision-making, yet none were gathered in such an organized manner. **This list provides a cornerstone that allows the next research steps to progress.**

4.2. Abstract

The research carried out to date shows that architectural decision-making is far from being a rational process. Architects tend to adopt a satisfying approach, rather than looking for the optimal architecture, which is a result of many human and social factors. The results of a workshop, carried out with 14 software engineering practitioners show that cognitive biases are commonly present in architecture decision-making. A systematic approach to analysing the influence of biases on decision making has been introduced. Twelve cognitive biases identified during the workshop were analysed with regard to the elements of the decision-making context that affected the aspects of architectural decision making. Finally, we analyse the interactions between cognitive biases and the conditions of real-world software development.

4.3. Introduction

The concept of architectural decisions enables the design rationale and architectural knowledge to be captured, but equally important is the focus it places on the act of deciding on software design [6]. This has shaped anew our perception of software development as a decision-making process, and triggered research into its nature.

The research on how architectural decisions are made revealed that, despite the intrinsic complexity of architecting, architects as decision makers remain normal human beings: their judgement is more often bounded rational than fully rational, which is a result of the inherent properties of the human mind.

As normal human beings, architects are subject to cognitive biases [105]. This exploratory paper, which has been motivated by the outcomes of a work-shop carried out with 14 software engineering practitioners, investigates how cognitive biases influence architectural decision-making. Our research focuses on answering the following research questions:

- Are biases in architecture decision making commonly observed by software engineering practitioners?
- What are the most significant cognitive biases that influence architecture decision making?
- Which of these biases result from cognitive biases inherent to the conditions of the human mind?
- Which elements of the decision-making context can bias architects' decisions?
- Which aspects of architectural decision making are influenced by the biases that have been identified?

- How do practical conditions influence the extent of the influence of the biases on architectural decision making?

The research presented in this paper shows that biases are commonly observed by software practitioners (Section 4.5). In order to capture how cognitive biases influence architectural decision making, we propose a model comprising contextual factors that are transformed by the biases into influence on the identified aspects of the architectural decision-making process (Section 4.5.2). The identified cognitive biases are presented and discussed according to this model (Section 4.5.2). The ways in which the practical conditions of architecture decision making affect the 'mechanics' of biases' influence is presented in Section 4.6. The discussion of the findings (Section 4.8) is presented in Section 6, the summary of the paper and the research outlook in Section 4.9.

4.4. Related Work

The social and human factors in software engineering have been studied since the advent of software engineering as a scientific discipline – compare reports from NATO conferences from 1968 and 1969 [106], [107]. Probably the earliest observation of the influence of social factors on software architecture is the now famous Conway's law [107].

The research on cognitive biases was pioneered by Tversky and Kahneman (e.g. [105], [108], [10]), and the latter was awarded a Noble Prize. Their research has been later extended by many researchers, e.g. [109], [42], [110]. Kahneman and Renshon [111] define cognitive biases as "predictable errors in the ways that individuals interpret information and make decisions".

The dual process theory, as stated by Kahneman [105], is one widely known and accepted in psychology and helps to explain the mechanisms of the human mind. According to the theory, our thoughts are controlled by two parallel systems: System 1 and System 2.

System 1 controls the part of our mind that is fast, runs effortlessly, and completely out of our control. It is very useful, letting us, for example, instantly react to danger and save our lives in a dire situation. Sadly, its associative nature may make our line of thought illogical, often creating associations where there are none. System 1 is unable to process rule-based logic, and thus does not always perform well.

System 2 is the complete opposite of System 1. It is slow, requires a great amount of conscious effort to use, and is very logical. Since the use of System 1 is something unavoidable, System 2 serves the purpose of correcting the premature conclusions of System 1, but only if we put effort into it. It is when this correction does not happen, or is not strong enough, that cognitive biases occur.

The process of architectural decision making has been thoroughly investigated by Zannier et al. [112] in 2007. Their research shows that architects use naturalistic (looking for a satisfactory solution) or rational decision making (looking for an optimal solution). The naturalistic approach is common for poorly-structured problems, while the rational one for well-structured problems.

In 2015, Tang and van Vliet observed [113] that most architects need only a few reasons before concluding a decision, which is a result of satisficing behaviour of naturalistic decision making.

The research record on the role of biases in software engineering in general, and in software architecture, is rather small. Tang and van Vliet, in 2016 [19], indicate only a couple of papers that can be related to the role of biases in design processes. They show some examples of anchoring, framing and confirmation biases in software engineering, and conclude that "biases do play a role; and we probably cannot fully prevent them from occurring; they are simply too human."

The research presented in this paper aims to expand upon these observations, and to systematise the way we analyse how cognitive biases influence architectural decision making.

4.5. Investigating Biases in Architectural Decision Making

4.5.1. Workshop on Biases in Architecture Decision-Making

Fourteen software engineering practitioners with a wide variety of professional experience (8 novices – 1-2 years of experience and 6 experts with at least 10 years of experience) were gathered on a workshop. The purpose was to gather data about biases that may have an influence over the process of software development and sort out those that affect architecture-decision making. A few days before the workshop, participants were provided with a list of 105 cognitive biases with their definitions and examples that purposefully were not connected to software architecture. They were supposed to get acquainted with them as a form of preparation. The workshop agenda was as follows:

- Short presentation on cognitive biases – the notion of 'cognitive bias' was explained and most important biases, such as Anchoring [10], the Bandwagon Effect [109], the Dunning-Kruger Effect [114] and the Law of the Instrument [115], were discussed;

- Poll of the participants – they were asked to write down any examples of bias-es that they could have observed from their previous projects;
- Survey and an open discussion on biases indicated by the participants, aimed at improving the understanding of people's statements;
- Rating of the biases - each participant was asked to rate the frequency of cases when they experienced the effect of every cognitive bias listed (on a scale from 0 to 3, where 0 means something that was never observed, and 3 means an often experienced phenomenon);
- Identifying related cognitive biases;
- Wrap-up and conclusion.

4.5.2. Influence of Cognitive Biases on Architecture Decision-Making

Expanding on the workshop results, we discussed and analysed the identified biases in order to identify how they influence the decision-making process. As a result we developed the model presented in Fig. 1.

Fig. 1 shows that cognitive biases subconsciously influence the decision-making process by associating some contextual factors with a given architectural problem. These factors usually have no connection, or just a limited connection, with the analysed issue itself. The biases include these factors into the decision-making process and alter the critical aspects of the decision-making process. Analysing the biases on the basis of our experience we identified sets of:

- *contextual factors* affecting architecture decision making, for example: form of presentation, architect's beliefs, "topics of the year", time spent on a given design, order, etc. – for a complete list see Table 4.2, Column (3).
- *aspects of architecture decision-making process* affected by the biases, for example such architect's preferences (expressed by a decision's rationale), scope of considered architectural issues, etc. – for a complete list see Table 4.2, Column (4).

Below we describe and study in a greater detail the cognitive biases indicated in Table 4.1.

Framing Effect. The example that achieved the highest average rating in the experiment was representative of the framing effect. The bias itself happens when information is judged differently on the basis of how it was presented. This effect was examined thoroughly by Tversky and Kahneman [108]. In the course of their research, they discovered that slight differences in the formulation of the choice problem may significantly impact decision making.

Table 4.1. Biases relevant to architecture decision making indicated by the workshop participants

Reported Bias	Related cognitive bias	Average Frequency Assessment
Judging the quality of a system by the form of its presentation by marketing specialists.	Framing effect	2.58
Estimating the time needed to complete a task wrongly, because of the expectations placed on the development team by the client, rather than the true complexity of the problem.	Confirmation bias	2.33
Excessive overvaluation of a solution that we created ourselves.	IKEA effect	2.33
Spending long hours on meetings discussing trivial problems (like whether we should use spaces or tabs in our code) instead of truly important ones.	Parkinson's Law of triviality	2.25
Insisting on continuing using certain COTS despite the long record of errors	Anchoring	2.25
Errors created by miscommunication between the technical staff and a client, because of them having a completely different background.	Curse of knowledge	2.16
Focusing mainly on a set of standards, believing that if they are met, then the quality of the product will be good.	Anchoring	2.16
Belief that "new is better". Quick abandonment of old tools and technologies while they are still properly working.	Pro-innovation bias	2.08
Dismissing serious issues as trivial, without any further thought.	Parkinson's Law of triviality	2.08
Evasion of solutions that were not used previously in our industry.	IKEA effect	2.00
Underestimating a problem by assuming that it could be solved by "writing some code", even with no detailed plan of the implementation.	Planning fallacy	1.91
Wrongly assuming that a solution found on the web is correct and appropriate for our problem, especially if it is popular.	Bandwagon effect	1.83
Avoidance of redoing work on a system that was already done – even if it was done poorly.	Irrational escalation	1.75
Being unable to admit that there is an error. Logic similar to "it's not a bug, it's a feature".	Anchoring	1.75
Applying the design patterns that we know everywhere - even if it's not the best solution.	Law of the instrument	1.66
Underestimating the possible load put on a system. Guessing without any evidence to support our claims.	Optimism bias	1.66
Focusing only on hardware when solving optimisation problems.	Anchoring	1.36

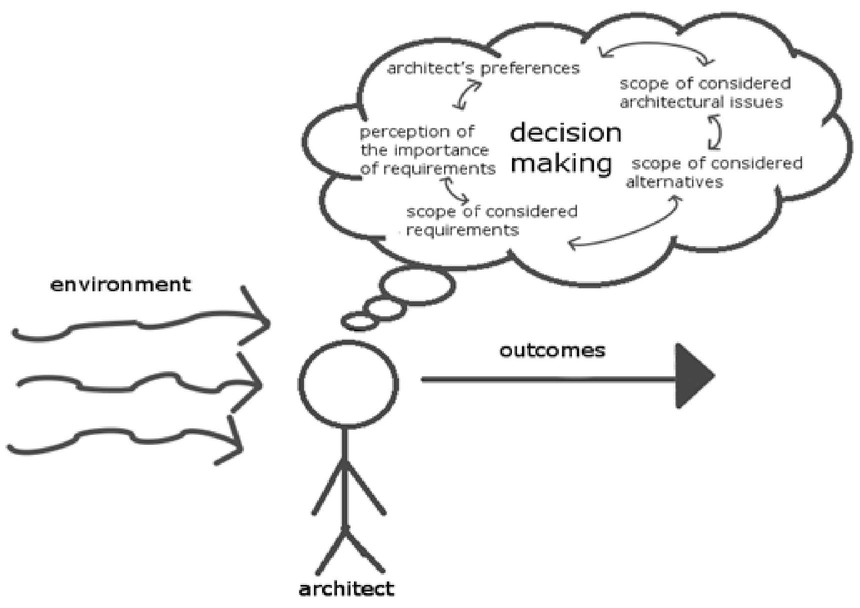
architect's preferences
scope of considered architectural issues
perception of the importance of requirements
decision making
scope of considered alternatives
scope of considered requirements
environment
outcomes
architect

Figure 4.1. How biases influence architecture decision making

The exact scenario that the workshop participants pointed at was that software products are often purchased not on the basis of their quality, but of the way they are advertised, for example: properly advertised products (like COTS) are more likely to be chosen, even if they are less suited to the needs of the project. The framing bias affects mainly architect's preferences, scope of considered architectural issues and alternatives.

Confirmation Bias. Another widely appearing case is an example of conformation bias. Confirmation bias itself, as stated by Nickerson [116], is a natural tendency to look for evidence supporting our claims, because we believe that this is an effective way of showing what is right, if it really is right.

The subjects linked the wrong estimation of the time needed to complete a project with their superiors' (or clients') expectations. Concerning architectural decision making, it is easy to observe a tendency to look for arguments confirming our beliefs that certain solutions are better than others. Believers of micro-services would always find evidence to confirm it is a best choice. In order to confirm their beliefs, architects may narrow the scope of considered architectural issues, requirements and alternatives. Such an architect would prefer to choose the options, which comply with his beliefs.

IKEA Effect. Easily associated with the phenomenon of the globally-known producer of ready-to-assemble furniture – the IKEA effect is one that makes us biased when it comes to items we have created or assembled ourselves [117]. What makes the furniture bought in IKEA so special, is that when you assemble it yourself, the false belief that it's worth more than you paid for it is created. All because you had to put your own time and effort into it. A similar effect can be observed in almost every domain, not only the furniture business. When comparing to the products of our competitors, even if they are renowned professionals in their field, we are easy to prey for the unstoppable gut feeling that our creation is the best.

The participants also pointed to the reverse form of this effect – that we tend to avoid solutions that have not been yet used in our industry, thus boxing our-selves in a small pool of alternative choices. It is a popular phenomenon that authors of a given system (application etc.) are rather reluctant to replace it even if a definitely better one is currently available. This will certainly narrow the set of considered alternatives and obviously affect architect's preferences.

Parkinson's Law of Triviality. According to Parkinson's Law: "work expands to fill the time available". This unavoidable effect rules over most workplaces, although most managers would prefer to overlook it. Parkinson's Law of triviality is a narrower version of Parkinson's Law stating that we spend an enormous amount of time debating over trivial unimportant issues [118].

Most of our subjects had the unfortunate displeasure of taking part in meetings that seemed to be endless and led to nothing. Although wasting time does not have to lead to wrong decisions by itself, it does shorten the time available to resolve more complex issues. This may result in 'system 1' being used to resolve the crucial problems instead of 'system 2'. This may indirectly affect all the aspects of architectural decision making. Anchoring.

The effect of anchoring is created by the way in which the human brain estimates probabilities. Naturally, we tend to cling to the first fact that comes to our mind when contemplating an issue. This means that the first piece of information we obtained, or one that we have a particularly fond memory of, heavily influences decision making [10]. When 'anchored' to an idea, it becomes hard to notice different solutions, and even if we do, it is very unlikely that we will choose them.

Although examples of the anchoring effect were rated very differently by the participants, it is worth noting that this was the bias for which they found the greatest amount of examples. Anchoring seems to have an influence over almost every kind of decision: hardware, technologies, design and even implementation. It influences mainly architect's preferences but also possibly the scope and perception of importance of requirements.

Curse of Knowledge. We may be put in a situation when we have to communicate with someone of a completely different background, or with a different level of experience, or even simply someone younger than us. We fall prey of the curse of knowledge, if at some point we falsely assume that the other side possesses the same knowledge that we have about any kind of issue. This may be more apparent in contacts be-tween children and adults, when the young ones find it hard to grasp concepts that are new to them, but adult relations are not free from this effect [110].

The curse may cause misunderstandings at any level of human interaction, which is especially crucial when understanding the requirements of our client and choosing appropriate solutions for their problems. Team members with different experience levels can also be influenced when an issue is not explained properly, they can misunderstand the way in which their tasks should be handled. Therefore, the curse of knowledge bias influences the scope of requirements and the architect's perception of their priorities, as well as scope of considered architectural issues and alternatives, and as a result this may also alter architect's preferences.

Pro-innovation Bias. The false belief that innovation should always be adopted is what we call the pro-innovation bias [119]. Humans naturally have a positive attitude associated with innovation. However, it does not mean that innovation should always be pursued.

The pursuit of a novel solution is not always necessary; sometimes it may even be harmful. What needs to be taken into account is the high risk of every innovative project. If a stable and reliable system is to be created in a reasonable time-frame, usually innovation should be avoided. As one of the subjects pointed out, it is not always the case that potentially high rewards await those that successfully bring new ideas to life. Relating these observations into architectural decision making, we observe that pro-innovation bias means that innovative design alternatives are considered and preferred.

Planning Fallacy. The planning fallacy is the tendency to underestimate the time required to complete a task. Interestingly, it seems to affect the individuals who are supposed to complete the tasks more than observers – of course, if they have information about the individuals' past performance [42].

This cognitive bias has a great influence over the planning phase of any project. The amount of information needed to avoid it is so big that it is almost impossible to process it, especially if a problem is complex.

Although various methodologies have their ways of soothing this problem (e.g. Planning Poker in Scrum), there are almost none that can prevent it on the early decision-making level. Let us also observe that the planning fallacy may affect the preferences of an architect who may choose solutions which are more difficult to implement than he thought when making a decision.

Bandwagon Effect. The bandwagon effect is a universal phenomenon that appears in almost any domain where human beings are given a choice. People naturally want to wear, buy, do, consume and behave like their fellows, thus becoming part of a group [109]. This results in popular choices and popular decisions becoming even more popular.

The danger this bias poses should not be downplayed – it puts our mind in a small box, limiting our possibilities and potentially forces bad decisions on us. Especially in cases of pressure from higher-ups to solve a problem in the way they wish us to. In some cases even, due to the organisation culture in a company, it may even be impossible to have any influence on this kind of decisions, which could result in choosing solutions being very far from perfect. Therefore the bandwagon makes an architect to prefer the same solutions that have already been widely accepted by a similar organisations, by an industry branch or our community.

Irrational Escalation. Irrational escalation takes place when one continues to commit to an initial course of action, even if it is obviously no longer the most beneficial choice [120].

As the participants noticed, this may affect performance when old technologies, code or components that are unfit for the task, are forced on us. Often these old products should have been abandoned long ago due to their doubtful quality, but since at some point they were invested in, there is a stubborn reluctance to let go of them. Irrational escalation means that architect gets fixed on an existing solution.

Optimism Bias. We do not usually assume failure before trying something. Most healthy people's brains are hardwired that way [121]. When writing his example, one of our participants told us a story where a client he worked for judged how many mes-sages his system would have to handle daily. Unfortunately, in reality, the estimated value turned out to be a hundred times too low, which triggered later multiple serious issues. As this example shows, being overly optimistic can hurt badly not only planning, but the architected system's quality as well. Architect's optimism can potentially influence all the aspects of architectural decision-making.

Law of the Instrument. Known in software architecture as the Golden Hammer anti-pattern [115] - when a single technology or design pattern is used in every possible place. This obviously results in the creation of numerous inefficient and mismatched solutions. The bias experienced here may simply be a symptom of the lack of necessary skill or knowledge that forces us to use well-known solutions. Such cases were pointed out by the participants of the workshop.

Furthermore, there is one more scenario that requires further consideration – does spending money on a technology in the past force us to use it? This seems to be the case in many big companies, where decisions to invest in expensive technologies are often made independently of the technical context of specific projects.

The above findings have been summarised in Table4.2. Note that in Column (3), only the most important factors influencing the decision making have been listed. Naturally, there are numerous other factors that may modulate (magnify or diminish) the influence posed by a give cognitive bias, for example the architect's knowledge/experience, organisation's culture

Table 4.2. Influence of cognitive biases on architecture decision making

Bias (1)	Description (2)	Main contextual factors that influence architect's judgement (3)	Influenced aspects of architecture decision making (4)
Framing effect	Drawing different conclusions from the same information depending on the form of presentation [108]	Form of presentation	Architect's preferences, scope of considered architectural issues and alternatives
Confirmation bias	Focus on searching for facts that confirms one's beliefs, while ignoring opposing information. [116]	The architect's beliefs	All the aspects.

IKEA effect	Overvaluing items that were created or assembled by us personally. [117]	Who was the author of a given design; time spent on a given design.	Scope of considered alternatives, preferences.
Parkinson's Law of triviality	Focusing time and effort on trivial matters while often omitting the truly important ones. [118]	None.	All the aspects.
Anchoring	Relying on one piece of information more heavily than any other, usually on the first one that we were exposed to. [10]	Order of obtaining information.	Scope and perception of importance of considered requirements; preferences.
Curse of knowledge	When individuals, due to having a different level of knowledge, interpret facts differently.[110]	The knowledge, experience and background of the stakeholders.	All the aspects.
Pro-innovation bias	An overly optimistic approach in adopting innovative solutions. [119]	The architect's state of mind with respect to innovative solutions.	Preferences, scope of considered alternatives.

Planning fallacy	Underestimation of the time it will take to complete a task. [42]	Problem complexity	Preferences
Bandwagon effect	The phenomenon of people more likely adapting ideas/buying products that have already been widely accepted. [109]	Existing widely accepted solutions.	Preferences
Irrational escalation	Continuing an action/investment, because of a similar prior one, even if the previous one turned out to be a wrong decision. [120]	Existence of an initial solution, course of action contradicting the use of an initial solution.	Preferences
Law of the instrument	Using a tool/skill that you possess everywhere, even in contexts where it is not appropriate. [115]	Architectural solutions focal for an architect.	Scope of considered alternatives, preferences.

Optimism bias	Overestimating the probability of favourable outcomes of our decisions. [121]	The architect's state of mind.	All the aspects

4.6. Cognitive Biases in the Practical Conditions of Architectural Decision Making

Having established that cognitive biases are a common phenomenon in architecture decision making, we explore real-world factors that can influence their manifestation or affect the magnitude of their influence on architecture decision making.

Biases and Time. Cognitive biases are an integral element of human nature. They have been shaped by the evolution of the human mind, and as such are a result of the adaptation to the conditions of the environment. Although they possibly distract architects from crafting a fully rational, thoroughly deliberated design, they potentially enable the qualities desirable by today's hectic software industry: rapid architecting and quick response to changes or emergencies.

As the "need for speed" concerns more and more software engineers, the role of System 1 will certainly be increasing at the cost of diminishing the role of System 2. It means that even more decisions will be made intuitively without a thorough deliberation. This may substantially hinder the quality of a software architecture. At the same time, architecting efficiently under the pressure of time is something very desirable in the frenetic software industry, as well as in emergency cases.

It seems that there are two basic ways of addressing this challenge:

- Applying debiasing techniques — this seems to be generally difficult, as the main factor limiting rational judgement is the lack of time and other external pressures. In order to ensure rational decision making, we have to give architects more time to conclude a decision and to restrain the external pressures. This is in many cases impossible, as we have limited or no control over the conditions that are external to architectural decision making;
- Accepting "the rules of the game" (biases) and trying to exploit them to our advantage — this requires the development of techniques that lead to reasonable architectures under pressure of time. Hypothesising further, they could take the form of a specific training for architects, probably similar to those used by students preparing for programming competitions: this

training supposedly makes system 1 closer to system 2 with regard to algorithms and computer programming, as trained students decide at a glance which algorithms should be used in order to solve their exercise.

Biases and Teams. Applying group architecture decision making techniques has the potential to limit the influence of biases, as decisions are made by people with different mindsets. This justifies assessing an architecture by a group of stakeholders, such as in ATAM. At the same time, group decision making brings with it the risk posed by the 'law of triviality' bias.

Biases and Cultural Factors. Cultural factors may magnify or diminish the influence of cognitive biases. For example, in many cultures it is difficult for people to admit they cannot under-stand something or accomplish a certain design. This will certainly strengthen the curse of knowledge bias.

Biases and Tools and Methodologies. It is also important to recognise that cognitive biases introduce a feedback be-tween what we create and how it is created, i.e. what we create, what we know influences how it is created by us. This is exactly what most of the biases do – consider, for example, anchoring bias, irrational escalation and the law of instrument biases. Therefore, it is worth investigating how different software development methodologies, architecture decision-making techniques, software development tools etc. interact with cognitive biases and vice versa.

4.7. Results

RQ.1 Are biases in architecture decision making commonly observed by software engineering practitioners? It turned out that the participants commonly observe biases in deciding on soft-ware design. Both novices (less than 2 years of experience) and experts (more than 10 years of experience) in software engineering noticed biases. Novices indicated on average 1 bias each, experts about 4 biases each. Experts have a much broader experience than novices, which explains the observed difference.

RQ.2. What are the most significant biases in architecture decision-making? As a result of our research, we identified 12 cognitive biases that influence architecture decision making. The list of these can be found in Tables 4.1 and 4.2.

RQ.3. Which of these biases result from cognitive biases inherent to the conditions of the human mind? All these biases, concerning architectural decision making, indicated by the workshop participants and listed in Table 4.1, can be related to well-known cognitive biases.

RQ.4. Which elements of the decision-making context can bias architects' decisions? These identified elements of the decision-making context that bias architects' decisions are: form of presentation, the architect's beliefs, who was the author of a given design, the time spent on a given design, the order of obtaining information, the knowledge, experience and background of the stakeholders, the architect's state of mind, the problem complexity, the existing widely-accepted solutions, the course of action contradicting the use of an initial solution, and architectural solutions focal for an architect.

RQ.5. Which aspects of architectural decision making are influenced by the biases that have been identified? The above aspects of architectural decision making are: the architect's preferences (finally expressed by the rationale for a decision), the scope of the considered architectural issues, alternatives and requirements, and the perception of the importance of requirements (compare Section 4.5 and Table 4.2).

RQ.6. How do practical conditions influence the extent of the biases' influence on architectural decision making? Time, teams, cultural factors as well as tools and methodologies used for soft-ware development can affect the extent of the biases' influence on architecture decision making.

4.8. Discussion, Limitations

The volume of research on cognitive biases in software engineering is rather small (compare Section 4.4). Let us observe that our research confirms the findings of Tang and van Vliet [19], namely, that anchoring, framing and confirmation biases are among the most often observed by software engineering practitioners as influencing architectural decision making.

The contribution of this paper comprises:

- the proposition of a model of how biases influence architectural decision making, which enables a systematic, uniform analysis of various biases;
- the identification of 12 cognitive biases that influence architectural decision making;

- an analysis of how each bias affects decision making, by identifying the elements of the model mentioned above (elements of the context affecting decision making and aspects of the decision-making process influenced by each bias);
- an analysis and identification of real-world factors that can potentially influence the extent of the influence of biases on architecture decision making.

The obvious limitation of the presented results are:

- the number of workshop participants may influence the representativeness of the results;
- although the claims of Section 4.6 seem to be logically sound, the analysis of real-world factors is only exploratory, hence it requires empirical substantiation to strengthen the claims of Section 4.6.

To provide an environment that would, as much as possible, neutralize the effects of additional biases and mistakes from the participants, all of them were informed thoroughly about the topic of cognitive biases both before and during the workshop which is described in more detail in Section 4.5.

4.9. Summary and Research Outlook

Cognitive biases are commonly present in architecture decision making. By asking practitioners, we identified 12 cognitive biases that can be observed most frequently. In order to analyse their influence on architectural decision making in a uniform way, we have developed a model of how biases 'work'.

The common presents of cognitive biases is both virtue and vice. On one side, they enable rapid architecting by an intuitive resolution of the architectural is-sues, on the other, they may lead to sub-optimal solutions and in extreme cases to a design disaster. We can either try to accept and exploit them, or fight them. Probably, we need a kind of a decision-making approach that balances 'system 1' and 'system 2' decision making.

The further research outlook includes:

- obtaining a more statistically significant confirmation of the above results by interviews with a larger group of practitioners or by a broader industrial survey;
- investigating the interactions that may exist between the biases;
- developing techniques of using the knowledge about biases and their influence on decision-making process, in order to align the architecting process with the stakeholders' expectations;

- carrying out an in-depth analysis of each of the identified biases.

5. The Influence of Cognitive Biases on Architectural Technical Debt

Article title	The influence of cognitive biases on architectural technical debt.
Authors	**Klara Borowa**, Andrzej Zalewski, and Szymon Kijas
Publishing venue	International Conference on Software Architecture
Year	2021
MNiSW points	140
Contribution	Original research idea, research method design, data gathering, half of the data analysis (coding), most of the paper writing.

5.1. Preface

Technical debt is a term used to define various software-related constructs that are beneficial in the short term and may possibly be harmful in the long term. In particular, technical debt impacts two software quality attributes: maintainability and evolvability [103]

There are various types of technical debt, depending on software artifacts, e.g., code debt, requirements debt, test debt, and notably - architectural technical debt [122].

Architectural, technical debt, being this paper's focus, is considered the most dangerous, long-lasting type of technical debt [24] since there are no straightforward methods of repaying such debt [28]. Additionally, architectural design decisions can be a source of architectural technical debt [25].

Having discovered what cognitive biases are most likely to impact architectural decision-making (Chapter 4), the next step was to find a specific real-life impact that is negative.

This paper presents an interview study performed with 12 practitioners. This study found which biases (optimism, anchoring, and confirmation bias) are the most important when it comes to causing unnecessary and dangerous architectural technical debt. Additionally, the antecedents of bias occurrence, as well as the consequences of the technical debt items described by the participants, were reported in this study.

This chapter's purpose is to show how dangerous cognitive biases' impact on architectural decision-making can be. In this case, the consequences of harmful architectural technical debt were explored.

Because of cognitive biases, architects may incur architectural technical debt unintentionally, which may severely harm the system's maintainability and evolvability. **A debiasing intervention should make this less likely to occur and, as such, improve the quality of the designed architectures.**

5.2. Abstract

Cognitive biases exert a significant influence on human thinking and decision-making. In order to identify how they influence the occurrence of architectural technical debt, a series of semi-structured interviews with software architects was performed. The results show which classes of architectural technical debt originate from cognitive biases, and reveal the antecedents of technical debt items (classes) through biases. This way, we analysed how and when cognitive biases lead to the creation of technical debt. We also identified a set of debiasing techniques that can be used in order to prevent the negative influence of cognitive biases. The observations of the role of organisational culture in the avoidance of inadvertent technical debt throw a new light on that issue.

5.3. Introduction

Technical debt is a metaphor first introduced by Cunningham [123] in order to explain the need of refactoring to non-technical stakeholders. It represents experiences that are common to many contemporary software system developers. Namely, the need to compromise between software quality (esp. internal) and other non-technical requirements, such as time-to-release/market. Despite the intense research undertaken to date, the mechanics of the process by which software technical debt arises is still far from being fully explained. This hinders the development and application of systematic technical debt management approaches.

The main factors that produce the above research challenge are:

- the substantial variety of types of technical debt (testing [124], source code [125], architectural [126], etc.);
- the variety of factors that contribute to the creation of technical debt [127] [125];
- the social and psychological nature of the phenomenon of technical debt [128] and
- the intrinsic complexity of the mechanisms underlying the creation of technical debt, resulting from its nature.

Technical debt, whether taken on deliberately or inadvertently, is always rooted in human thinking and/or its limitations. Cognitive biases are an important factor that shape human thinking and decision-making [14], often distorting its results, making decisions diverge from those fully rational ones. As a result, it is important and necessary to analyse the influences of cognitive bias on the emergence of technical debt, as this aids an understanding of how technical debt arises, and leads to the development of efficient management strategies.

This paper presents our research on the influence of cognitive biases on the occurrence of architectural technical debt (ATD). It was focused on the following research questions:

- **RQ1.** Do cognitive biases influence the occurrence of architectural technical debt?
- **RQ2.** Which cognitive biases have an impact on architectural technical debt?
- **RQ3.** Which architectural technical debt items are most frequently affected by cognitive biases?
- **RQ4.** What are the antecedents of a harmful influence of cognitive biases on architectural technical debt?
- **RQ5.** What debiasing techniques can be used to minimise the negative effects of cognitive biases?

In order to answer the above questions, we performed semi-structured interviews with 12 architect-practitioners and analysed their outcomes. Detailed information on the research methods can be found in Section 5.5, which is preceded by an overview of the current state of research in Section 5.4. The research outcomes are presented in Section 5.6 and discussed in Section 5.7. The threats to validity are discussed in Section 5.8 and the outcomes summary and further research outlook is presented in Section 5.9.

5.4. Related Work

Cognitive biases, originally observed by Kahneman and Tversky [33], [11], are a phenomenon inherent to the human mind. They are rooted in the duality of the human reasoning process – according to Kahneman and Tversky – there are two systems responsible problem resolution that exist and operate within human mind. System 1, which is responsible for quick intuitive decisions based on a limited scope of information. System 2, which is suited for logical and fully rational reasoning on the basis on a broader set of information. System 2 is invoked consciously whenever we analyse and rationally resolve problems. If we do not consciously and carefully consider our decisions (i.e. by employing rational thinking of System 2), System 1 will draw premature conclusions that will not get corrected by System 2. In such case, we can say that a cognitive bias has influenced our reasoning and its outcomes.

The possible influence of cognitive biases on Software Engineering has been already a topic of interest for researchers for over two decades [52]. Cognitive biases may possibly influence any Software Engineering activity [14], be it requirements engineering [51], design [129], development [20] or testing [55]. Architecture decision-making [130] is also not excluded from the impact of cognitive biases [22], [77], [26].

The core thesis of this paper, whereby cognitive biases, by distorting the decision-making process, may contribute to taking on technical debt, is at an early stage of research. Only papers [20], [128] [127] indicate that such an influence is possible, though there have been no detailed investigations on this topic.

Technical debt has been a topic of extensive study in the recent years. It has resulted in a huge number of papers, including systematic literature reviews [28], [131], [132] and even a tertiary study [133] summarising the state of research on technical debt.

Architectural technical debt (ATD) is the type of TD that occurs as a result of sub-optimal architectural decisions [24]. ATD can be especially dangerous since it may hinder the development of future software features [134]. According to Ernst et al. [135] – "architectural issues are the greatest source of technical debt." The purpose of this paper is to expand the existing knowledge of how architectural technical debt arises by analysing how cognitive biases contribute to its emergence.

5.5. Research Method

5.5.1. Cognitive biases

The number of cognitive biases that possibly can have an influence on software development is vast [14]. As it is not feasible to analyse every cognitive bias in a single study, we decided to focus on the biases listed as the most relevant for architecture decision-making in the exploratory work of Zalewski et al. [26]. In this work, the researchers attempted to elicit which cognitive biases have the most significant impact on architecture decision-making. These are:

- *The framing effect* – the tendency to judge information and make decisions based on the how the data is presented [136].
- *Confirmation bias* – this effect influences individuals that have a strong belief they do not want disconfirmed. As such, they search only for information confirming this belief, and ignore any proof that they may be in the wrong [116].
- *Anchoring bias* – This bias occurs when one's judgement is strongly influenced by the first piece of information given to them. [137] Thus, it often results in individuals having an irrational preference for the first solution/idea that they came up with or heard about from someone else.

- *Curse of knowledge bias* – this cognitive bias manifests itself in experts that consider part of their knowledge as obvious, which then results in miscommunication when they interact with other people [138].
- *IKEA effect bias* – is the irrational preference for solutions that have been at least partially developed (or assembled) by ourselves [139].
- *Parkinson's Law of triviality bias*– when a disproportionately large amount of time and effort is put into performing trivial tasks and solving trivial problems [140].
- *Pro-innovation bias* – the assumption that innovation is a value in itself. Which means that new solutions should always be adopted everywhere, as soon as possible [141].
- *Planning fallacy bias* – the tendency to underestimate the time necessary to complete a given task [142].
- *Bandwagon effect bias* – the desire to "join the crowd" and do what others do [143]. This means that popularity becomes the main factor taken into account when choosing between options.
- *Irrational escalation bias* – the irrational impulse to continue wasting resources on an investment that is not cost-effective [144].
- *Law of the instrument bias* – sometimes referred to as the "law of the hammer" since when you own a hammer, everything seems to be a nail [145]. This law states that we tend to overuse tools and solutions that we already own or are familiar with.
- *Optimism bias* – the unjustified belief that in our case, in the same scenario, we are more likely to obtain a positive outcome than others[146]. This effect makes individuals more liable to make risky decisions, despite evidence that it may not be reasonable.

5.5.2. Architectural Debt items

Having prepared a list of biases worth exploring, we also had to specify the kinds of architectural technical debt to be researched. There are different approaches to categorising ATD [147], [148], but for this purpose we decided to use the architectural technical debt items defined by Verdecchia et al. [127] as the most commonly occurring. We hoped that, since this categorisation emerged from gathering data during interviews, it will also be easily understood by our participants. Those items [127] include:

- *Re-inventing the Wheel* – which manifests itself when we use a self-developed component rather than a stable, verified one that is easily available.

- *New Context, Old Architecture* – which occurs when not enough effort is put into keeping the evolution of the architecture appropriate for its context.
- *The Minimum Viable Product (MVP) that stuck* – which appears when software that was hurriedly developed for a simple temporary solution ends up becoming part of larger system that is still evolving. Architectural gaps of the MVP solution are inherited by the system.
- *The Workaround that stayed* – which appears when a temporary workaround is used in order to sidestep architectural constraints. However, it becomes deeply ingrained in the system and is never removed.
- *Architectural Lock-in* – which occurs when a component is so deeply embedded into the system that replacing it would be extremely expensive or even unworkable.
- *Source Code architectural technical deb*t – a type of ATD that has its source solely in the implementation of the solution.

5.5.3. Research procedure

The research method assumed in this paper follows the guidelines for case studies in software engineering by Runeson et al. [98]. In order to investigate the influence of cognitive biases on the occurrence of architectural technical debt, we decided to carry out an empirical enquiry based on a set of semi-structured interviews with software architecting practitioners.

The general outline of the interview process that we developed for the purpose of this study, and employed during each of the twelve interviews with architects, was as follows:

1. The interviewer asked the participant for their consent to record the interview and to use the acquired data for research purposes.
2. The researcher obtained statistical data about the participant (age, gender, years of experience, position, company size/domain).
3. The interviewer introduced the participant to the topic of technical debt and our research.
4. The participant was provided with the definition of each architectural debt item. Then they were asked if they had ever encountered this type of technical debt and, if so, whether they could describe their experience with it. This is the part of the interview in which the participants had the freedom to provide any information that they wanted and believed to be relevant.
5. The interviewer asked the subject if they had any other experiences with technical debt that they had not mentioned yet.

We did not suggest, either before or during the interviews, that cognitive biases may influence technical debt. The participants were informed in advance that we were researching reasons for the occurrence of technical debt, but we could not disclose which reasons we were researching, in order to avoid influencing their answers. After the interviews, if the participant was interested, we disclosed information about our research on cognitive biases. In some cases, this resulted in obtaining additional insights, which were written down for further analysis.

Almost all of the interviews were conducted in Polish, with the exception of No. 7, conducted in English.

5.5.4. Study participants

Table 5.1. Participant data

No.	Age	Gender	Experience (years)	Position	Company size (employees)	Company domain
1	29	M	5	Software Developer	over 10 000	Electronics
2	31	M	10	Architect	around 2 000	E-commerce
3	54	M	35	Chief Operating Officer	around 1 500	High tech
4	37	M	13	Executive consultant	around 50	Systems integrator
5	39	M	17	Head of Architects	around 350	Finance
6	49	M	26	Architect	around 350	Finance
7	37	M	16	Consultant	over 10 000	Enterprise Software
8	45	M	21	Chief of Architects	around 250	Systems integrator
9	36	M	15	Founder and Chief Technology Officer	around 35	Software
10	37	F	15	Architect	around 5 000	Telecom
11	40	M	15	Senior Solution Architect	over 10 000	Enterprise Software
12	37	M	12	Team Leader	over 10 000	Electronics

In order to find architects-practitioners, we created an advertisement which we propagated using our private networks. Most of the participants currently work as architects, though some also had prior architecting positions and now worked as leaders/managers/company owners. We also interviewed one software developer, which provided us with a valuable distinct point of view. The overall data about the participants is summarised in Table 5.1.

Table 5.2. Qualitative analysis codes

Code category	**Code**	**Definition**
Cognitive Bias	CB: [bias name]	Occurrence of one of the cognitive biases from the list in Section 5.5
Architectural technical debt occurrence	ATD: [item type]	Occurrence of technical debt, which can be classified into one of the architectural technical debt items mentioned in Section 5.5. This code was to be used only in cases when the participant gave a real-life example of a technical debt occurrence.
Architectural technical debt occurrence	ATD: Other	Unclassified occurrence of architectural technical debt
Architectural technical debt occurrence influenced by a cognitive bias	CB influencing ATD: [note]	Cases when a cognitive bias directly resulted in the creation of technical debt. The note should contain details of which bias influenced what kind of technical debt items and how.
Cognitive bias influencing factor	CB antecedent: [note]	Antecedents of the appearance of cognitive biases. Note should contain a further description.
Debasing methods	Debiasing: [note]	Information about interventions that were suggested or performed by the participants, which could result in a debasing effect.

5.5.5. Analysis Procedure

Having obtained the raw data from the interviews, we performed the analysis in the following steps.

1. The recorded interviews were transcribed.
2. We created a coding scheme for analysing the data, using the guidelines of Runeson et al. [98]. This codes are presented in Table 5.2.
3. Each of the authors encoded the transcripts independently.
4. Using the negotiated agreement [101] approach, we discussed the coding and incrementally corrected it until we reached unanimity.
5. The following metrics were extracted from the transcripts: the number of cases of cognitive bias mentioned by the participants, occurrences of architectural debt, cases of cognitive biases influencing architectural technical debt. Those are presented in Tables 5.3, 5.4 and 5.5 in Section 5.6.
6. The factors influencing cognitive bias as well as the debiasing methods mentioned by participants were extracted. They are presented in Section 5.6.

7. The notes from the interviewer were analysed, in search of any additional data that should be taken into consideration while drawing the conclusions.
8. The results were discussed and conclusions drawn in a discussion between the authors.

5.6. Results

5.6.1. Architectural debt items influenced by cognitive biases

The participants provided us with accounts about their previous projects, in which various architectural debt items could often be observed simultaneously. While analysing the interview transcripts and interviewer notes, we identified 70 specific occurrences of architectural technical debt items, the exact number of occurrences of each ATD item is shown in Table 5.3.

Table 5.3. Technical debt occurrences mentioned by participants

Architectural technical debt item	Appearances
New Context, Old Architecture	17
Source Code ATD	13
The Workaround that stayed	12
Architectural Lock-in	10
Re-inventing the Wheel	8
The Minimum Viable Product that stuck	6
Other (4 different types of ATD)	4

Despite providing the participants with the definitions of architectural technical debt and the specific architectural debt items, they often mentioned situations that were not cases of architectural technical debt, or which fit the definition of a different technical debt item. A common mistake was the false belief that the use of an old technology is synonymous with technical debt, which does not have to be the case.

The most commonly occurring ATD item was "New Context, Old Architecture". This specific kind of technical debt appears naturally, by itself, over time, if not enough effort is given to periodically update, upgrade, change or refactor the architecture. This classical problem has already been described in one of Lehman's laws of software evolution [149], which says that at some point of time, perpetually evolving systems reach a threshold when it is no longer cost-effective to evolve further without carrying out a major system's reconstruction. Many of our participants observed, that this moment often passes unnoticed.

We found several instances of technical debt that did not exactly fit any of the categories of ATD items specified by Verdecchia et al. [127]. These technical debt items are:

- *Choosing an obsolete solution that should not be used in the current circumstances at the start of the project.* Participant No 5 explained that in his company, decision-makers only consider aged solutions as "safe enough" to be used.
- *Reusing a component in a setting, in which it does not fit the given problem.* Participant No 1 told us how one of his colleagues focused on using readily-made solutions so much, that it resulted in choosing a solution completely unsuitable for their problem.
- *Using a proven architectural solution in a new context, in which it is not suitable.* Participant No 4 explained a situation in which they had to deal with data on the client's customers, for which they used a readily-made component that integrated all the aspects of every customer's data. The problem appeared when it turned out that not all the customers wanted to have all of their data connected to a single account, since they may want to create many accounts for various purposes.
- *Transfer of organisational debt onto architectural technical debt.* Organisational debt occurs when key decisions (such as writing down contracts, defining strategies or assigning responsibilities) are not made in time. This, in turn, may affect key design decisions, which have to be made with incomplete data about the problem at hand. Participant No. 3 gave us an example of a situation when a state-owned system had to be deployed before certain key political decisions were made. This resulted in the need to redo the basic components of the system.

We did not observe a correlation between the participants' experience, position or their company's domain and the number of ATD items they observed. An interesting case of this came from participants Nos 5 and 6. Both worked in the same organisation, No 5 had nine years of experience less than No 6, but participant No 5 gave us nine examples of ATD item occurrences, while participant No 6 mentioned only two.

5.6.2. Cognitive biases that influence ATD items

By analysing the transcripts and interviewer notes, we found 155 occurrences of cognitive biases. The exact numbers for each bias are shown in Table 5.4. It was not unusual for many biases to influence a single ATD item, which often occurs consecutively in a cascade of irrational decisions, such as:

- A decision-maker heard that a specific technology is popular, which led him to believe that it may be useful in his case (Bandwagon effect)
- He met with a salesman of this specific solution, who only informed him about the beneficial aspects of the solution, which persuaded him to buy it (Framing effect)
- Despite the disadvantages of this solution, it was used simply because it had already been paid for (Irrational escalation)
- Which led to an "Architectural Lock-in" because this component was so specific and deeply embedded in the system that it turned out extremely difficult to replace.

In our analysis we took into account not only the biases of the architects and developers, but also those that influenced other stakeholders involved in the development and maintenance process – such as the management and the clients.

Most of the biases mentioned as possibly crucial in the work of Zalewski et al. [26] had a notable influence on ATD. An exception here is Parkinson's law of triviality. This specific bias may impact the time spent on certain tasks, but it does not seem to significantly change the final outcomes, and so has little magnitude when it comes to causing ATD.

Table 5.4. Cognitive biases present in the participants' accounts

Cognitive bias	Appearances
Anchoring	24
Bandwagon effect	8
Confirmation bias	19
Curse of knowledge	14
IKEA effect	14
Irrational escalation	11
Law of the instrument	10
Optimism bias	20
Parkinson's Law of triviality	2
Planning fallacy	10
Pro-innovation bias	13
The framing effect	10

5.6.3. Influence of cognitive biases on ATD items

In this Section we discuss in-depth how particular ATD items were impacted by cognitive biases. Table 5.5 presents the exact number of times that a certain cognitive bias influenced particular ATD items. The data contained in this table does not add up to the data from Table 5.4 and Table 5.3, because a specific bias occurrence may have influenced a single ATD item more than once, and one ATD item may have been influenced by more than one cognitive bias.

If the influence of a particular bias on a specific ATD item was reported at least three times, we explored thoroughly the relationship between that bias and the ATD item.

Table 5.5. Cognitive biases influencing ATD items

Cognitive Bias	**New Context, Old Architecture**	**Source Code ATD**	**The Work-around that stayed**	**Architectural Lock-in**	**Re-inventing the Wheel**	**Minimum Viable Product that stuck**	**Other**
Anchoring	7	5	4	6	4	1	0
Bandwagon effect	0	1	1	1	1	0	0
Confirmation bias	2	2	5	4	5	1	1
Curse of knowledge	2	2	2	4	2	0	0
IKEA effect	3	3	1	2	3	1	0
Irrational escalation	7	1	2	0	1	1	0
Law of the instrument	1	3	2	3	0	0	0
Optimism bias	3	3	3	5	2	4	1
Parkinson's Law of triviality	0	1	2	0	0	0	0
Planning fallacy	3	4	4	3	1	1	1
Pro-innovation bias	1	1	2	4	4	2	1
The framing effect	1	2	2	3	1	1	0

1. *New Context, Old Architecture*

   The significant impact of cognitive biases on architectural technical debt can be clearly observed when researching the causes of this ADT item.

   Four participants experienced a situation when a solution was chosen simply because it was the first possible one that came to notice (*anchoring*), and even though it was not cost-effective and did not enable the further evolution of the product, resources were persistently being wasted on it (*irrational escalation*).

   Participant No. 1 for example, was involved in a project where an open source solution was chosen to create a simple dashboard for the end user. Unfortunately, as the solution evolved and expanded, the source code of this component had to be forked. Ultimately, the participant's team introduced an enormous amount of changes and became the maintainer of this newly created solution. This increased the team's workload with the maintenance efforts.

Too often, a component was used after the support for it had expired, which either left the component without maintenance or forced the client to take the path of expensive, dedicated individual support. Participant No. 11 told us about a few cases of systems made for the public sector, when his clients needed this kind of individual maintenance. This could have been prevented by properly preparing for the time when this problem was bound to occur, but often no such precautions are taken, and decisions are made without long-term planning. In general, it seems that the decision to start from scratch with completely new architecture is made reluctantly. This hesitancy can be motivated by many cognitive biases: the *IKEA effect* when the old solution was made by the decision-maker's organisation (participant No. 9 called a product his "precious business baby"), or the *optimism bias* which may make the decision-makers believe that no harm may come to them (for example, when using a system that is not maintained properly).

2. *Source Code ATD* Source code ATD is most often influenced by *anchoring*. When solving a problem, the very basic, satisficing approach is widely present. Citing participant No. 11 – "If something stupid works, then it is not stupid." This is especially apparent in organisations in which decisions on how to implement certain components are left entirely to individual developers whose ideas are never challenged. This was the case in an example provided by participant No. 2, who had the displeasure of "inheriting" a completely unscalable solution, used as a basis for a key e-commerce platform developed by his company.

   This problem of satisficing decision-making is strengthened by other cognitive biases as well. The *IKEA effect* makes developers choose (or even copy) from their own previous work, the *law of instrument* makes them use only tools that they are familiar with. The *confirmation bias* makes them blind to information that their decisions may be wrong, an effect that is often enhanced by the *optimism bias*.

   Participant No. 12 encountered a combination of all of this biases in the form of an enormous Bash based solution, used to deploy changes to the production environment. The author of the solution was comfortable with Bash, and did not consult (nor was he challenged) his solution's design with anyone. He did not take into account that anyone else may ever need to read, understand or change his code. This resulted in the creation of an enormous set of Bash scripts that were extremely hard to comprehend to anyone besides its author.

All these problems become even more relevant when not enough time is given for thorough consideration, which is an intermediate effect of the *planning fallacy* – when the planned time for tasks was too short.

3. *The Workaround that stayed*

   A substantial number of workarounds generally come from two beliefs. Namely, that there is no choice, and that fast fixes are a normal and proper way of problem solving. Individuals that firmly believe in either of them often do not put any additional effort into considering the, often lacking, rationale behind their "fixes" (*confirmation bias*).

   Having come up with an idea for a simple workaround, they are satisfied that the problem will be promptly resolved, and do not search for any alternatives (*anchoring*).

   Participant No. 3 gave us an interesting example, of an organisation that routinely used a complicated set of workarounds while processing accounting data. Only when a new integrated accounting system was introduced, the organisation did realise that they have been producing faulty financial reports for the last 5 years. This kind of a mindset is further strengthened when they are not given enough time, because the the need for such such tasks has not been foreseen (*planning fallacy*).

   However the workaround does solve the immediate problem, so there is no urgent need to change the state of things. This approach, heavily tainted by the *optimism bias*, was displayed by Participant No. 9 with the words "we will hopefully come back to it one of these days."

4. *Architectural Lock-in*

   The most common pattern behind the occurrence of the Architectural lock-in ATD item was a combination of *anchoring* and *optimism bias*. Firstly, due to anchoring, the first satisficing architectural solution was chosen. Then, even though they did not have any experience with the solution, development teams simply took a leap of faith (*optimism bias*) and used the solution without further consideration of whether it may be difficult to replace later.

   This would not have become such a serious problem if individuals did not have a tendency to choose risky innovative solutions (*pro-innovation bias*), or if they took time to consider the disadvantages of their architectural concepts (*confirmation bias*).

Participant No. 12 provided us with the following example that illustrates this problem. The maintenance team in his organisation needed a ticketing system. They found an simple open source solution on GitHub that looked satisfactory and choose it, blindly believing that it would be the proper one. They did not take into account that the component may require changes in the future and that it was PHP-based (no one in that team had prior experience in PHP). When they needed to expand the solution, they found themselves "locked-in" this particular component, while not having the skills necessary for its further development.

Additionally, we noticed that decision-makers often relied on data from salesmen, which is of course always prepared in a way that shows the offered solutions in a positive light (*framing effect*). Participant No. 3 particularly stressed how "salesmen should never be trusted". Even if decision-makers attempted to obtain information from experts within their own organisation – the experts had a tendency to omit key information during meetings, because of the false belief that such knowledge is obvious (*curse of knowledge*).

5. *Re-inventing the Wheel*

   The Re-inventing the wheel ATD item was mainly observed by our participants in the context of younger, inexperienced team members, as well as in small companies, especially start-ups. Lacking prior experience, and with the possibility to start something new from scratch, ambitious young people often fall into the temptation of creating a solution they would have full control over (thus, *anchoring* on that single aspect), something they could call "their own" (*IKEA effect*). They want to be pioneers (*pro-innovation bias*), despite often not possessing and not searching for already existing knowledge. This frequently results in re-inventing the metaphorical wheel.

   Participant No. 7 told us about a case when he worked in a small company, with a colleague that he described as an "IT geek". This coworker developed a web application framework on his own. He frequently applied it when creating products for the company. After he changed job, this undocumented framework was left without its core maintainer.

   The unwillingness to face the reality that someone may have already had the same idea and properly put it into effect, and thus the inability to find and use ready-made components, is a symptom of the dangerous influence of the *confirmation bias*.

6. *The Minimum Viable Product that stuck*

   We found that the MVP that stuck ATD item was the least likely to appear from the list defined by Verdecchia et al. [127]. In our participants' accounts, MVPs rarely "got stuck", which means that our observations for this type of ATD were limited. In the case of most MVPs mentioned by our participants, these solutions were either abandoned as prototypes when a superior solution was found/developed, or these MVPs were expanded and matured over time.

For an MVP remaining as it is over an extended period of time, someone had to make a mistake while estimating a very short lifespan for the product (*optimism bias*). Participant No. 5 told us that this usually happened when small programs were written in a hurry to perform simple tasks like processing/converting text files or uploading/downloading them.

5.6.4. Cognitive bias antecedents (RQ4)

Having identified the biases that influenced the generation of technical debt, we made an in-depth analysis of the participants' accounts in search of information on why these cognitive biases occurred. Cognitive biases, as a phenomenon inherent to the human mind, cannot be completely avoided. However, certain factors can amplify the influence of cognitive biases on architecting activities and their outcomes. These factors, namely, the antecedents of cognitive biases, can be divided into seven groups:

1. *Individual's emotional state*

   We found that certain feelings often precede the appearance of biases. These are:

   a) Fear of: change, responsibility, consequences and of starting from square one (precedes *anchoring, confirmation bias, irrational escalation*);
   b) Shame, especially of one's past mistakes (precedes*anchoring, irrational escalation*);
   c) Feeling a lack of agency (makes individuals less likely to challenge the ideas of others and provide a debiasing effect);
   d) Haste (makes individuals more susceptible to*all* biases, especially *the planning fallacy*);

   Fear and shame have a notably destructive effect. One of our participants attempted to register information about the technical debt in their organisation. This turned out to be difficult because employees, even managers, were unwilling to share information about the technical debt they were responsible for. They were afraid of consequences and ashamed of their previous mistakes, which then hindered the process of actively managing technical debt. Individuals experiencing great fears are unlikely to change their behaviour, which makes them even more susceptible to biases such as *anchoring, confirmation bias and* *irrational escalation*.

Feeling a lack of agency is especially relevant in large organisations, in which individuals often feel that they have no influence on any decisions that are being made. Because of that, they simply remain indifferent, do not challenge others' decisions, and end up mindlessly following orders.

2. *Individual's personality traits*

   There were two kinds of personality types that were overly prone to cognitive biases. The extremely ambitious and confident individuals and their opposite – the reserved ones that lacked assertiveness. The overconfident architects tended to make fast decisions without deeper consideration (this makes *all* cognitive biases more likely to appear), while the taciturn team members tended to follow them blindly. In this way, the possibility of exerting a debiasing effect on their colleagues is lost. It also makes them more prone to the *bandwagon effect*.

3. *Individual's mistakes*

   We observed some common mistakes that foreshadowed the appearance of cognitive biases. Those included:

   - The basic lack of knowledge or experience required to make decisions in a certain area (if sufficient knowledge is not obtained, *any* cognitive biases are more likely to occur).
   - Not performing any search for alternative solutions (*anchoring, confirmation bias, IKEA effect, law of the instrument, pro-innovation bias*).
   - Considering only a limited part of a complex problem, while ignoring the global impact of the solution (*anchoring, confirmation bias, curse of knowledge, optimism bias, planning fallacy*).
   - Limiting the planning only to the short term (*planning fallacy, optimism bias*).
   - Habits (*confirmation bias, anchoring, irrational escalation, IKEA effect, law of instrument*).
   - The optimistic belief that mistakes are only made by others (makes *all* biases more likely to appear).

   An interesting detail is that even seemingly good habits, such as using proven architectural patterns, may turn out to be harmful. In the case of this particular habit, sometimes a design pattern may end up being used in unsuitable circumstances.

4. *Organisational antecedents*

The overall environment in which the project is being developed and maintained has a crucial impact. We observed several factors that impacted the frequency of bias appearance:

- Organisational culture: too lax (which strengthens all biases in individual employees) or too harsh (which impacts the biases of management and leaders).
- Frequent changes of management staff that impedes long-term planning (influences *all* biases)
- Lack of standards and procedures (*all* biases).
- Unclear separation of duties, especially when it is not clear who is responsible for which decisions (this means that nobody provides a debiasing effect when decisions are made in this area).
- Short-sighted cost/profit optimisation as a default approach – investing only in areas that give immediate profit (*irrational escalation, optimism bias, anchoring*).
- Lack of motivation for optimising the developed solutions – especially in the case of short-term cooperation with clients (optimism bias).
- Faulty use of agile development practices – empowered by the belief that any problems can be fixed in further iterations (influences *all* biases, since decisions are made in each iteration by all the parties involved in the project)

A valuable observation that we made was that, in most of the interviews, cognitive biases appeared as a consequence of an organisational culture that was either too lax or too harsh. When the culture in the organisation was too casual, which is often the case in startups (participant No 9 provided us with such insights), individuals are often left to make key decisions alone. If these decisions are influenced by cognitive biases, no one challenges them and thus various faulty decisions are made – mainly by young, overambitious team members. This may lead to many biased decisions like choosing trendy solutions (*bandwagon effect*) or using only tools that the decision-maker is familiar and comfortable with (*law of instrument*). On the other hand, if the organisation's culture is authoritarian, giving little voice to the employees in lower positions in the hierarchy, then possible biases of the higher-ups are never challenged or corrected. In such cases, decision-makers are more susceptible to the enticements of salesmen (*framing effect*), or do not have information that would allow them to plan the time-frames for projects properly (*planning fallacy).*

5. *Communicational antecedents*

Many biases emerge as an after-effect of communication problems. This most commonly happens when specialists from different domains interact, although even close co-workers are not free from this problem. These problems are often fuelled by the *curse of knowledge*, which makes individuals more likely to omit crucial information that could be obtained from others.

Constructive criticism and challenging the ideas underlying the decisions of others is not standard in every team. It often means that crucial decisions are never discussed openly. This means that valuable debiasing opportunities are lost, which in turn makes biased decisions more likely to occur. Additionally, sometimes decisions made during the initial negotiation phase of a project are made without consulting technical specialists, which leads to overly strict deadlines and sometimes absurd contractual arrangements. Participant No 11 told us that, in the case of projects made for the state, they commonly found that the price and time-frame for the project were made absurdly low and short during negotiations, simply in order to gain the customer, in hope that profit could be increased later and fixes could be made during the maintenance phase.

6. *Knowledge vaporisation*

   For any decision to be rational, it is essential for the decision-makers to have proper knowledge about the issue at hand. However, it is a well-known issue in the area of architectural knowledge management that knowledge is not always documented and tends to vanish with the employees that leave the company. This makes decision-makers more prone to the effect of *all* cognitive biases, since they are frequently forced to make decisions based solely on their instincts.

7. *External*

   Some antecedents to cognitive biases are completely beyond our control. The most prevalent is the current popularity of certain solutions – the deciding factor behind the *bandwagon effect*.

   Although, often forgotten, an important source of possible problems also lies in politics and the current legal status. A proper interpretation and understanding of the law is not an easy task and often leads to dangerous misconceptions. One of our participants provided us with an account in which they had to create a system before a particular law came into force. This law had been incorrectly understood by the developers (the *curse of knowledge* influenced their communication with legal specialists), which resulted in the need to perform a set of quick fixes and workarounds shortly after the system became available.

Frequent changes in the law, influenced by politics, may also force numerous technically challenging modifications to existing systems. *Every* such decision is susceptible to the influence of cognitive biases – in this case, the politicians' biases.

5.6.5. Possible debiasing methods (RQ5)

During the interviews, our participants spontaneously gave us hints, how to avoid arising certain ATD items. Additionally, the participants that scarcely encounter specific ATD items, usually mentioned why they believe that this item does not occur often in their environment. All this combined together, in many cases, can be interpreted as a set of possible debiasing techniques.

We gathered this into a set of bias prevention treatments:

1. Ensuring double-checking and challenging all decisions and their underlying ideas. Trying to find downsides of any idea as standard – this will make the critique feel less personal and is therefore more effective.
2. Developing an environment based on trust, in which employees can voice their opinions and admit to their mistakes – knowing that they will receive help, not scorn. The modern approaches to agile / servant leadership address this issue.
3. Explicitly gathering information about alternatives before making decisions. Presenting them to others and asking for their evaluation. This is reminiscent of some components of architecture evaluation methods.
4. Creating procedures and standards to limit low-quality reckless alterations of the solution and enable periodical refactoring/changes of the system to fit the ever-changing business context in which it is used.
5. Creating documentation and passing on knowledge. At a bare minimum, this does not require much resources, it might involve recording meetings in which information is shared and decisions are made.
6. Explicitly registering all accounts of TD in the organisation and making plans for a time when it will be dealt with.
7. Periodically checking whether any new TD has occurred and whether any old TD needs to be paid – maybe support expired, or all the people with the relevant knowledge have moved on and are no longer present.
8. Clearly defining and recording who is responsible for which part of the project's scope. This

will make the process of obtaining information and looking for help more straightforward. It will also minimise the problem of the individuals in charge avoiding responsibility.

5.7. Discussion

As the presented results have no direct equivalents (the possible influence of biases on technical debt has been merely mentioned [128],[127], [20]), it is necessary to relate these results to broader research on technical debt, cognitive biases in SE, antecedents and management techniques for technical debt.

Firstly, we partially confirmed the findings of Zalewski et al. [26], since almost all the biases (with the exception of Parkinson's law of triviality) that they recognised as notable for architecture decision-making, were found to have had a significant influence on ATD. Furthermore, the biases that we have identified as most commonly influencing ATD (anchoring, optimism and confirmation bias) have already been identified as having a significant influence on software engineering activities [14].

In the field of architectural decision-making, it has already been noticed that cognitive biases distort the decision-making process [23] by strengthening the effects of pre-existing problems/mistakes. As such, the antecedents for cognitive biases are bound to be, at least partially, similar to previously discovered causes of ATD. The problems of time pressure, lack of documentation, unsuitable architectural decisions and human factors, as specified by Verdecchia et al. [127], are similar to many of the antecedents that we identified. The issue of miscommunication between specialists of different domains, and its influence on technical debt, has also previously been addressed [150].

The debiasing methods that we propose only affect debt created inadvertently, since debt deliberately taken on is usually a result of a rational management strategy [151]. The debiasing methods that we proposed give an interesting new perspective to the problem of managing architectural technical debt – they can be taken as a set of instructions on how to manage an organisation that would be less susceptible to ATD. ATD management so far, as indicated by Besker et al. [28], suffers from a lack of proper management guidelines. A set of strategies has only recently been proposed [127], [152].

5.8. Threats to Validity

As with every study, certain issues might pose a threat to the validity of our findings. Having this in mind, we attempted to minimise the effects of such threats. Since the research is qualitative, and our goal was to conduct an exploratory in-depth analysis, we only took into account the experiences of our 12 participants. To prevent this from being a problem, we attempted to make this group as varied as possible – we interviewed people that held different positions, had varying levels of experience (from 5 to 35 years), worked in companies of different sizes (from start-ups to large corporations), and whose organisations had diverse domains.

Still, in order to further confirm the validity of this research, it would be useful to expand it with more participants, and possibly using a different research methodology (like the think aloud protocol method [153]).

Since our participants were not experts in the field of technical debt, they often presented examples of cases that were not actually occurrences of technical debt. Furthermore, even if their example was indeed a case of technical debt, they confused various ATD items withe each other. To ensure that such mistakes did not have an undue influence on our results, we searched for the ATD items (coded them from the transcriptions) without taking into account the ATD item category that the participant believed their example belonged to.

Since it may be possible for a single researcher to make a mistake during the coding – for example, to observe a cognitive bias that did not actually occur – the interview transcriptions were analysed and coded by us separately, and then the findings were confronted using the negotiated agreement approach [101]

To prevent our participants from forcibly searching for cognitive biases in their experience, we only asked them to explain the rationale behind the decisions made in their projects. They were informed about our cognitive bias related research only after the interviews.

Finally, cognitive biases often overlap and interact with each other. Which means that their influence on ATD items may not always be straightforward. We did not analyse the dynamics of bias' interaction in this paper.

5.9. Conclusion and Research outlook

Our research achieved the following:

- We assessed that cognitive biases definitely have a significant influence on the creation of architectural technical debt (RQ1).

- We determined that most significant biases that impact architectural technical debt are anchoring, optimism and confirmation bias. Nevertheless, the influences of the curse of knowledge, the IKEA Effect, irrational escalation, pro-innovation bias, planning fallacy, the framing effect and the bandwagon effect are also noticeable.
- We assessed that cognitive biases affect all of the ATD items indicated by Verdecchia et al. [127]; nevertheless, the most frequently affected item turned out to be "New Context, Old Architecture" (RQ3) and the least influenced one appeared to be "the MVP that stuck".
- The most common antecedents of cognitive biases that influence ATD have been identified (RQ4).
- A number of debiasing techniques have been proposed (RQ5).

Our research revealed also that the organisation's culture is often an important factor that influences the creation of technical debt, since most of the antecedents and the discovered debiasing methods are connected with how the organisation is managed, and the frame of mind of the organisation's members.

In order to minimise the amount of unwanted architectural technical debt, organisations should remove the fear of admitting software deficiencies and introduce trust into the company's culture. As noticed by Besker et al. [152], the penalising approach to managing architectural technical debt is the least effective one.

An atmosphere of thrust and camaraderie would enable individuals to provide a debiasing effect to each other. The ideal environment would be one in which challenging the ideas of others is commonplace, in the form of sensible, non-judgemental critique. If meaningful critique of each other's ideas becomes commonplace, employees are less likely to feel threatened by it and to actually start making use of each other's suggestions.

In an organisation founded on trust, there should be space to admitting one's mistakes. When a problem is detected, everyone should focus on solving it together, instead of looking for scapegoats.

Further research could include:

- further confirming our findings with proper quantitative data;
- how team / organisational culture influences the emergence of inadvertent technical debt;
- in-depth research on particular antecedents' and biases' influence on ATD;

6. Is knowledge the key? An experiment on debiasing architectural decision-making - a pilot study

Article title	Is knowledge the key? an experiment on debiasing architectural decision-making-a Pilot study.
Authors	**Klara Borowa**, Robert Dwornik, and Andrzej Zalewski
Publishing venue	Product-Focused Software Process Improvement: 22nd International Conference,
Year	2021
MNiSW points	70
Contribution	Original research idea, research method design, half of the data analysis (coding), most of the paper writing.

6.1. Preface

As shown in Chapters 4 and 5, cognitive biases can negatively impact software architects' decisions in various ways. One such effect is making architects unnecessarily incur dangerous architectural debt. Chapter 5 results have an additional important finding: that the main cognitive biases impacting architects are optimism bias, anchoring, and confirmation bias.

The natural progression of this research was to attempt to alleviate the impact of cognitive biases on architectural decision-making. In this pilot study, we performed a simple B level [78] debiasing intervention. We recorded 2 groups of students who were designing and implementing a system as part of their university curriculum. One of the groups took part in a simple presentation where they were informed about cognitive biases and how they may impact their project's architecture.

We recorded two separate project meetings for both groups and compared the arguments they used during their group discussions. In this case, the intervention turned out to be ineffective - the group that took part in the workshop used more biased arguments during their conversations than the non-debiased students.

However, this study had a major merit. We analyzed the conversations to find how exactly each bias impacted the team. The three researched biases (anchoring, optimism bias, and confirmation bias) were found to interact with each other – an effect which, in this thesis, is called the "**wicked triad**."

In accordance with these findings, proposed a set of debiasing techniques. These were:

- Listing drawbacks by the person proposing a solution: since usually, the first mention of a solution resulted in anchoring on its benefits.
- Listing risks: since optimism bias and confirmation bias mainly impacted the overall atmosphere (not the architectural arguments) of the meeting, making team members unlikely to consider any risks.
- Assigning somebody to monitor the meeting for the "We already decided on that, why discuss this more?" argument: This argument was usually a sign that confirmation bias was stopping the participants from discussing solution alternatives.

The development of the above debiasing techniques would not have been possible without this pilot study. Chapter 7 presents their experimental validation.

6.2. Abstract

The impact of cognitive biases on architectural decision-making has been proven by previous research. In this work, we endeavour to create a debiasing treatment that would minimise the impact of cognitive biases on architectural decision-making. We conducted a pilot study on two groups of students, to investigate whether a simple debiasing presentation reporting on the influences of cognitive biases, can provide a debiasing effect. The preliminary results show that this kind of treatment is ineffective. Through analysing our results, we propose a set of modifications that could result in a better effect.

6.3. Introduction

The occurrence of cognitive biases is inherent to the human mind, and as such, can influence all individuals taking part in the software development process [14]: developers [20], architects [22], designers [15], testers [154].

In particular, cognitive biases have been proven to distort architectural decision-making [77] by influencing software architects' reasoning [22]. This influence can be particularly strong, since every systems architecture is actually a set of design decisions [155] made by individuals. Thorough education about cognitive biases turned out to significantly improve software effort estimation [29], which is severely afflicted by cognitive biases [16]. Similarly, in this work we examine, *(RQ) whether educating software architects about cognitive biases can provide a beneficial debiasing effect, which increases the rationality of decision-making*.

In order to answer this question, we designed an experiment and ran a pilot study on two groups of students. The preliminary findings show that educating engineers about the possible impact of cognitive biases is not sufficient to mitigate the influence of cognitive biases on design decisions.

Therefore, more advanced debiasing techniques are needed. We analysed how exactly cognitive biases influenced various elements of the conversation (arguments, counterarguments, and general conversation). Based on that, we proposed additional debiasing techniques that can be used in order to create a more effective debiasing treatment. We plan to perform a modified version of this experiment, on a larger sample, in the near future. Our long time objective is to develop effective, debiasing techniques for architectural decision-making.

6.4. Related Work

The concept of cognitive biases was introduced by Tversky and Kahneman in their work about Representativeness, Availability and Anchoring biases [33]. Cognitive biases are a by-product of the dual nature of the human mind – intuitive (known as System 1) and rational (known as System 2) [11]. When the logic-based reasoning of System 2 is not applied to the initial decisions of System 1, we can say that the decision was biased.

Software architecture, defined as set of design decisions [155], is influenced by various human factors [75]. One of these factors are cognitive biases [77]. Their influence on architectural decision-making has been shown as significant in recent research [22] [77] [23] [26]. When no debiasing interventions are applied, the consequences of such biased decisions can be severe – for example resulting in taking on harmful Architectural Technical Debt [156].

In the domain of architecture decision-making, various debiasing techniques were proposed [77], [156]. The use of techniques that prompt designers to reflect on their decisions, have turned out to be effective in improving the quality of the reasoning behind design decisions [157].

Debiasing, by educating software developers about the existence of cognitive biases and their influences, has recently been proven to work as a powerful tool in the realm of software effort estimation [29]. The effectiveness of this approach to debiasing architectural decision making, has not yet been empirically tested.

6.5. Study Design

6.5.1. Bias selection

Based on the cognitive biases researched previously in relation to software development [14], as well as biases shown previously as influencing software architecture [26], [156], [77], we selected three cognitive biases as the subject of the experiment:

1. Anchoring – when an individual over-relies on a particular solution, estimate, information or item, usually, the first one that they discovered or came up with [33].
2. Optimism bias – when baseless, overly positive estimates, assumptions and attributions are made [45].
3. Confirmation bias – the tendency to avoid the search for information that may contradict one's beliefs [158].

6.5.2. Data acquisition

In order to obtain the data for our study, we took part in four meetings with two groups of students that were working on a group project during their coursework. The meetings were conducted online through the MS Teams platform. Both groups were supposed to plan, design and implement a system as a part of their course. The topic for the project was at their discretion, with the only hard requirement being the use of Kubernetes in their solution.

In the case of one of the groups, we prepared a presentation during which we explained the concept of cognitive biases, and how they can influence architectural decision-making. We explicitly explained the three researched cognitive biases and gave examples of their possible influence on the students' project. We did not mention anything about cognitive biases or debiasing to the second group.

The meetings proceeded as follows:

1. We asked the participants for their consent to record the meeting and to use their data for the purpose of our research.
2. In the case of the debiased group (Team 2), we showed them our presentation about cognitive biases in architectural decision-making. We did not perform this action with the other group (Team 1).
3. The meeting continued naturally, without our participation, although a researcher was present and made notes when necessary.

We also asked the participants to fill in a small survey to obtain basic statistical data about them.

6.5.3. Data Analysis

The recordings from the meetings were transcribed. In order to identify the cognitive biases, and their influence on decision-making, we defined a coding scheme presented in Table 6.1. The codes were applied to indicate the occurrence of the researched biases, as well as the arguments for and against the discussed architectural decisions.

The first and second author coded the transcripts independently. Then, they used the negotiated coding [101] method to discuss and correct the coding until they reached a full consensus.

Subsequently, we counted the number of occurrences of each code, and analysed the fragments of the meetings that were found to have been influenced by cognitive biases.

Code category	Code	Definition
Bias – Anchoring	KOT	Putting too much emphasis on the first piece of information or idea that was heard/proposed/invented.
Bias – Optimism	OPT	Naive faith that the unpleasant consequences of our decisions will not happen. Typical statements include: “It will somehow be.”, “No need to think about possible problems.”, “Let’s just start coding, it will be fine.”
Bias – Confirmation	POT	Not accepting and not seeking information that is inconsistent with our current beliefs.
Arguments for the decision	ARG	An argument that was in favour of choosing a particular solution.
Arguments against the decision	PARG	A counterargument, against choosing a particular solution.

Table 6.1. Coding Scheme

6.5.4. Participants

We recorded four meetings with two different groups of students that were working on their Master’s degrees in Computer Science at Warsaw University of Technology. The students grouped themselves into teams depending on their own preferences and had to choose a team leader. The teams consisted of five members each. Most of the students (with a single exception) had prior professional experience in software development. More detailed information on the students is presented in Table 6.2.

Age	**Gender**	**Has professional experience?**	**Job position**	**Experience [years]**	**Team No**
23	M	Yes	Data Engineer	1	1 (not debiased)
24	M	Yes	Software Developer	2.5	1 (not debiased)
23	M	Yes	Software Developer - intern	0.1	1 (not debiased)
24	M	Yes	Cloud/DevOps	3	1 (not debiased)
23	M	Yes	Systems Engineer	2	1 (not debiased)
24	M	Yes	Java Developer	1.5	2 (debiased)
24	M	Yes	Full Stack Developer	2	2 (debiased)
24	M	Yes	Java Developer	2	2 (debiased)
23	F	Yes	Sales Analyst	1	2 (debiased)
25	M	No	No professional experience	0	2 (debiased)

Table 6.2. Participant data

6.6. Results

Using the coding scheme presented in Table 6.1, we obtained the following information:

- The percentage of biased arguments in statements for or against certain architectural decisions (see Figure 6.1).
- How many arguments for and against certain architectural decisions were made during the meeting (see Figure 6.2).
- How many of these arguments and counterarguments, were influenced by cognitive biases (see Figure 6.3)
- How many cognitive biases were present in statements not related to architectural decisions (see Figure 6.3).

Figure 6.1, which presents the percentage of biased arguments used during the meetings, shows that Team 1 (non-debiased) used more rational arguments than Team 2 (debiased). This means that the debiasing treatment – simply informing the participants about the existence of cognitive biases – was ineffective.

Figure 6.2 shows that there was a significant difference between the amount of arguments and counterarguments in the discussions. Teams were less likely to discuss the drawbacks of their decisions than their positive aspects.

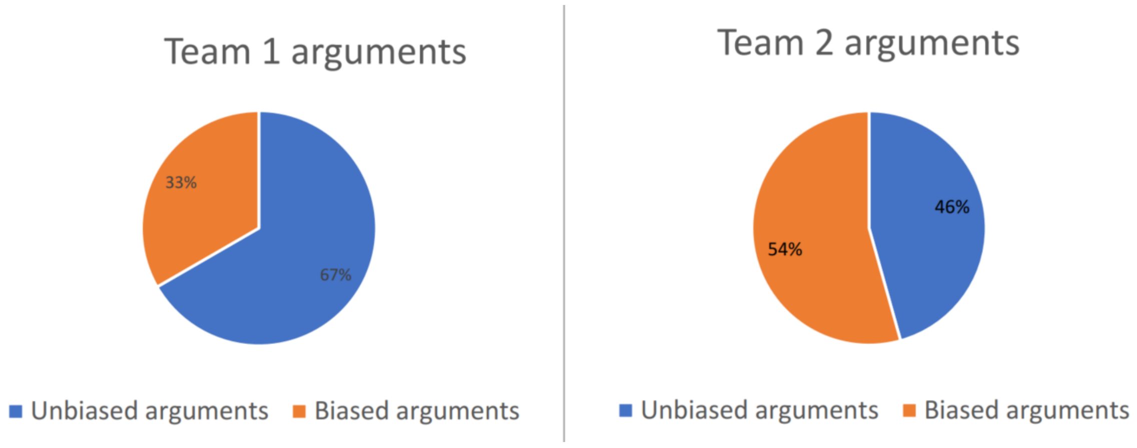

Figure 6.1. Biased arguments

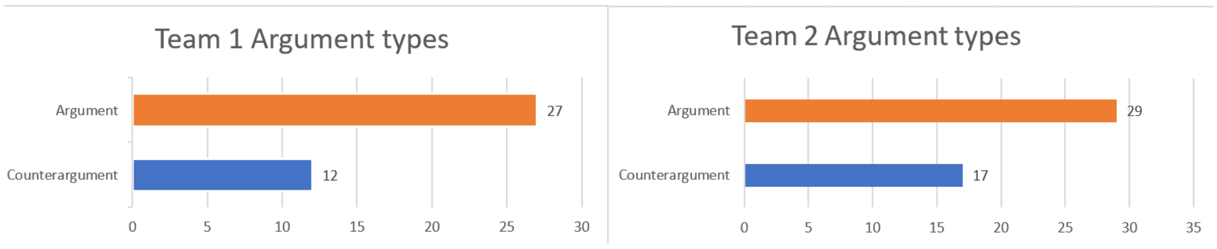

Figure 6.2. Argument count

Figure 6.3 illustrates the number of biased statements, as well as the ratio between the researched biases depending on statement type.

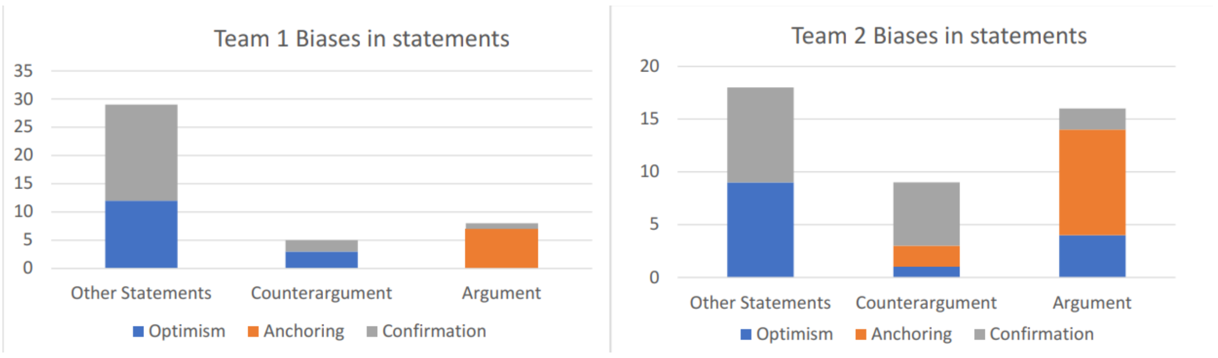

Figure 6.3. Biases in statements

In the case of both teams, most cognitive biases were present in statements not related to architectural decision-making. In this type of discussion, confirmation bias and optimism bias were the most prevalent. This was usually due to the teams' need to reassure themselves that their course of action was correct.

In both teams, most of the biased arguments were influenced by the anchoring bias. This means that both teams considered an array of solutions that came to their minds first, without any additional argumentation on why the specific solution is correct. When it comes to counterarguments, against specific architectural solutions, confirmation bias was prevalent in both teams. This was usually due to the teams' unwillingness to change a previously made decision.

6.7. Threats to validity

In this work, we describe a pilot study. Its main weakness is the small number of participants that took part in the experiment. This means that all of our findings are preliminary and cannot be perceived as final. We plan to perform a modified version of this experiment with a larger number of teams, to obtain more data to verify our findings.

6.8. Discussion

The team that was not debiased by our presentation used a significantly lower number of biased arguments. This implies that a simple debiasing treatment, by simply reporting on the biases is not strong enough to counter the influence of cognitive biases on architectural decision-making.

We discovered the typical scenario of bias-influenced architectural decision making. First, one team member proposes an idea that first came to their mind (an idea prompted by System 1). If the solution does not disturb the current project, other team members are unlikely to give any counterarguments (only around half of the arguments used were counterarguments) as they are already anchored on the initial proposition. If the solution requires changes to previously made decisions, other team members (due to confirmation bias), are likely to give biased counterarguments to avoid changes. Additionally, the whole atmosphere of the conversation is heavily influenced by the confirmation bias and optimism bias, making the team unlikely to notice any errors in their decision-making.

With these findings in mind, we propose (Section 6.9) a set of modifications to our debiasing approach.

6.9. Research outlook

Since the pilot study showed that a simple debiasing treatment does not help to overcome the biases, we plan to extend and repeat this experiment with the following modifications:

- Since the most biased arguments in favour of a solution were influenced by anchoring, and participants were overall less likely to use counterarguments – we propose that the person presenting a solution, should also present at least one drawback.
- Since most biased counterarguments were influenced by confirmation bias, due to the teams' reluctance to change a previously made decision – we propose that one of the team members should monitor the discussion and point out the occurrence of such a biased argumentation.
- Since optimism bias and confirmation bias influenced the overall atmosphere of the meetings – we propose that, at the end of the meeting, after making the initial decisions, teams should explicitly list their drawbacks. Then, if the need arises, decisions should be changed accordingly.
- We will add an additional code to the coding scheme - "decision". Which will mean the decision that was ultimately made during the meeting. This will enable us to count how many rational and biased arguments were made in favour of the decisions that were eventually chosen.
- Instead of a simple debiasing presentation, we will hold a longer debiasing workshop. During this workshop, we will do more than simply inform the participants about the

influence of cognitive biases on architectural decision-making. The participants will also be taught, through a series of practical exercises, how to apply our debiasing techniques.

- The next experiment will be performed on a significantly bigger sample of participants.

6.10. Conclusion

The preliminary results (see Section 6.6) show that a simple presentation about cognitive biases and their possible influence on architectural decision-making is not an effective debiasing method. At the same time the pilot study revealed crucial information about how biases influenced the arguments for and against certain decisions. This made it possible to develop a series of modifications to our debiasing approach (as presented in Section 6.9) in order to reshape the entire experiment.

7. Debiasing architectural decision-making: a workshop-based training approach

Article title	Debiasing architectural decision-making: a workshop-based training approach.
Authors	**Klara Borowa**, Maria Jarek, Gabriela Mystkowska, Weronika Paszko, Andrzej Zalewski
Publishing venue	European Conference on Software Architecture
Year	2022
MNiSW points	140
Contribution	Original research idea, research method design, part of data gathering, half of the data analysis (coding), most of the paper writing.

7.1. Preface

The unsuccessful debiasing intervention presented Chapter 6 has triggered the development of a more sophisticated, level C [78] debiasing intervention. This intervention has the form of a debiasing workshop during which participants are informed about cognitive biases and their impact on architectural decision-making. Finally, they take part in three practical exercises designed to teach them the debiasing techniques proposed in Chapter 6.

In this paper, we performed an empirical validation through a controlled experiment on student participants in order to evaluate this workshop. Twelve 2-3 person groups of master's students that took part in a software architecture course took part in this experiment. The overall plan of data gathering was:

- Phase 1: Designing an architecture as part of a given task (without debiasing),
- Phase 2: The debiasing workshop,
- Phase 3: Designing an architecture (different from Phase 1) as part of a given task (after debiasing).

Then, we compared the quality of the argumentation used by student groups before and after the debiasing workshop.

The experiment turned out to be successful, 10 out of 12 groups reasoning improved (became more rational) after the workshop. However, this was the case not because the number of biased arguments and counterarguments decreased - but because the number of non-biased statements increased. Additionally, contrary to our predictions, we found that the technique of "having a person monitor the meeting to counter confirmation bias" did not have a major impact.

The success of the debasing treatment presented in this study was a step towards the ultimate goal: debiasing experienced practitioners. As such, an additional verification on practitioners was performed and presented in Chapter 8.

7.2. Abstract

Cognitive biases distort the process of rational decision-making, including architectural decision-making. So far, no method has been empirically proven to reduce the impact of cognitive biases on architectural decision-making. We conducted an experiment in which 44 master's degree graduate students took part. Divided into 12 teams, they created two designs – before and after a debiasing workshop. We recorded this process and analysed how the participants discussed their decisions. In most cases (10 out of 12 groups), the teams' reasoning improved after the workshop. Thus, we show that debiasing architectural decision-making is an attainable goal and provide a simple debiasing treatment that could easily be used when training software practitioners.

7.3. Introduction

Cognitive bias is a term that describes an individual's inability to reason entirely rationally; as such, it prejudices the quality of numerous decisions [19]. Researchers have observed the influence of cognitive biases on day-to-day software development over two decades ago [158]. Since then, it was proven that almost all software development activities are affected by cognitive biases to some extent [14]. Architectural decision-making in particular is not exempt from the influence of cognitive biases [22], [19]. However, research on debiasing architectural decision-making is scarce [14], with a clear lack of empirically proven debiasing methods that could be used in practice [21]

In this paper, we endeavour to create an effective debiasing treatment, through expanding on our previous work [159]. The debiasing treatment that we designed consists of an hour-long workshop during which individuals learn about cognitive biases in architectural decision-making (ADM) and take part in three practical exercises. We tested the effectiveness of this debiasing treatment in an experiment, in which 44 master's level graduate students took part. Our study was aimed at answering the following research question:

RQ. Is a training workshop an effective method of reducing the impact of cognitive biases on architectural decision-making?

Through our study, we show that debiasing ADM is an attainable goal, since in most cases (10 groups out of 12) the debiasing treatment was successful. Our workshop provides a debiasing effect and, because of its simplistic design – it can easily be used to train software practitioners to make more rational decisions.

This paper is organised as follows. In Section 7.4 we describe research related to the subject of our study. Section 7.5 presents the research method, and in particular: the design of the debiasing workshop, our experiment, the study participants and how we analysed the obtained data. Section 7.6 contains the results of our experiment. In Section 7.7 we discuss our findings. The threats to validity are explained in Section 7.8. Finally, in Section 7.9 we provide a conclusion and describe possible future work.

7.4. Related work

Cognitive biases impact how decisions are made by every human being. In particular, they heavily influence intuitive decisions made under uncertainty [160]. This effect occurs due to the dual nature of the human mind, which comprises intuitive and rational decision-making subsystems [11]. Fischoff [78] describes four levels of debiasing (reducing the effect of biases) treatments: (A) warning about the biases, (B) describing typical biases, (C) providing personalised feedback about the biases, (D) an extended programme of debiasing training.

Cognitive biases influence on architectural decision-making. Tang [22] described how distorted reasoning may impact software design, by providing a set of examples of biased statements that software designers may use during their work [22]. As software architecture is actually a set of design decisions [6], it may be heavily affected by architects' biases. This makes reducing the impact of biased decision-making an important endeavour in the area of software architecture (see also [19], [26]).

Debiasing architectural decision-making. Although there are various guidelines and practices for improving architectural decision-making [74], [80], there is a severe lack of empirical research on treatments for undesirable behavioural factors in the realm of ADM [21]. There is a small amount of research on debiasing in Software Engineering. So far, the existing research has rarely proposed debiasing approaches, and empirical validation of the proposed debiasing methods [14] is even less frequent. Notably, Shepperd et al. [29] proposed a successful treatment that improved software effort estimation, through a two- to three-hour-long workshop about cognitive biases. Our team attempted an empirical validation of an anti-bias treatment for ADM [159], but it turned out not to be successful. This may be due to its several weaknesses:

1. We informed the participants about biases through a simple presentation. This treatment is on the lower levels (A and B) of Fischoff's debiasing scale [78]. In comparison, the

successful treatment proposed by Shepperd et al. [29] included a workshop and giving personalised feedback (level C debiasing).

2. In order to evaluate whether the treatment provided the desired effect, we compared the performance of two groups of students – one that was shown the presentation, and one that was not. However, this approach does not take into account the teams' individual traits. Those traits may make them more or less susceptible to cognitive biases from the start. It is possible that, when comparing a single team's performance before and after the presentation, the results may have been significantly different.
3. The sample (2 groups consisting of 5 students) was rather small.

This paper summarises our subsequent research that was aimed at developing a successful debiasing treatment by overcoming the above shortcomings.

7.5. Research Method

The three-hour-long experiment was performed during a meeting on the MS Teams platform. The experiment plan has been made available online [161]. While planning the experiment, we enhanced most steps from our previous approach [159] to both improve the debiasing treatment itself and the validity of the experiment. The basic steps of the experiment included:

1. Preparing the debiasing workshop.
2. Gathering participants.
3. A series of three-hour long meetings during which we conducted the experiment, which consisted of three steps:
    a) Task 1 – a 1 hour-long ADM task.
    b) The debiasing workshop.
    c) Task 2 – a 1 hour-long ADM task.
4. Analysing the teams' performance during the first and second tasks.

7.5.1. Biases

The debiasing workshop was designed to counter three biases that in previous research turned out to be exceptionally influential on architectural decisions [19], [162] and their impact on software engineering overall has already been researched extensively [14] :

1. Anchoring – a biased preference towards initial starting points, ideas, solutions [160].

2. Confirmation bias – when the currently desired conclusion leads the individual to search for confirming evidence, or omitting other information [26].
3. Optimism bias – an inclination towards overly optimistic predictions and judgements [14].

7.5.2. Architectural decision-making task

Each team of participants performed the task twice: before and after the debiasing workshop. The theme (the problem that was to be solved) was different in each task. The task was to design an architecture that could be used as a solution to a given theme, and to record the design using the C4 model notation [163]. The task itself was known to the participants before they took part in the experiment, in order to allow them to prepare and learn more about the C4 model. This was not the case for the themes. All the tasks were supposed to be graded as part of the students' software architecture course. However, the students were given over a week after the experiment to finish and polish their design. During both architectural design tasks, the researchers did not take an active part in the architecting.

7.5.3. Debiasing Workshop Design

The full workshop plan with instruction for workshop organisers have been made available online [161]. The workshop was designed to teach three debiasing techniques:

- The anti-anchoring technique: having proposed an architectural solution, the individual that presents it must explicitly list one disadvantage of the solution.
- The anti-confirmation bias technique: one team member has to monitor the discussion for unjustified statements that dismiss new information and ideas. Such as "We already decided that".
- The anti-optimism bias technique: the team must explicitly mention the risks associated with the design decisions.

These specific techniques were proposed previously as a result of our previous work [159] where we analysed, in detail, how each of the three researched biases usually impacted the teams that took part in the study. However, during that study, the effectiveness of these techniques was not validated. For each of these three techniques, the participants had to actively perform a practical exercise. In this phase of the experiment, the researchers actively facilitated the workshop by encouraging participants to use the debiasing techniques, providing them with examples of the techniques' use, and prompting the participants when they forgot to use the technique that they were supposed to apply.

7.5.4. Sample

The participants were recruited from among master's level graduate students majoring in Computer Science in our Faculty. These graduate students in particular, were taking a Software Architecture course. Although participation could be part of their graded project, it was voluntary. There was an alternative, traditional way, to obtain a grade. At the start of the MS Teams meeting, participants filled a questionnaire that allowed us to obtain basic data about them. Overall, 61% of the participants had prior experience in software development, ranging from 0.3 to 3 years. The questionnaire and its results, containing detailed information about the participants, is available online [161].

7.5.5. Analysis

For the analysis, we used a modified approach of our method from our previous study [159]. In order to analyse the results, we transcribed all recordings of the tasks, during which the participants' created their design. In order to inspect how biases impacted architectural decisions, we applied the hypothesis coding method [164]. This means that we defined a set of codes to be used to mark relevant segments in the transcript in advance, prior to the analysis. The coding scheme and all specific code counts have been made available online [161].

Each transcript was first coded by two researchers separately. Then, all of the codes were negotiated [101] until the coders reached a consensus on each code. Additionally, no transcript was coded by the same researcher that conducted the particular meeting with the participants. Furthermore, we summarised the overall number of codes only twice, after coding 6 and 12 transcripts, to avoid a situation where we would unconsciously chase after a desired number of biased or non-biased arguments in a particular transcript.

Having coded the transcripts, we compared how many biased and non-biased statements/decisions were present before and after the workshop. We defined: (a) a biased decision as one impacted by more biased statements than rational arguments, (b) a non-biased decision as one impacted by more rational arguments than biased statements, (c) neutral decisions as ones impacted by an equal amount of biased and non-biased statements. We also counted the amount of bias influences and the usage of the debiasing techniques during the tasks.

7.6. Results

Through the analysis process we uncovered the specifics about arguments, decisions, bias occurrences and the use of debiasing techniques in the teams' Task 1 and Task 2 transcripts. All p-values mentioned in this section were calculated using the non-parametric Wilcoxon Signed Rank Test. Through this test, we evaluated whether the changes in specific measured values were statistically different (when the p-value was less than 0.05). All specific numbers for code counts for each team are available online [161].

7.6.1. Arguments.

We classified these arguments as either biased (i.e. affected by one or more of the researched biases) or non-biased. Overall, we found 1470 arguments and 487 counterarguments. 54% of the statements before the workshop were biased, compared to 36% after. In general, the percentage of biased arguments decreased after the workshop in the cases of all teams except one.

The increased number of non-biased arguments (p-value = 0.0024) and non-biased counterarguments (p-value = 0.0005), and the decrease of the percentage of biased statements (p-value = 0.002) were significant. However, the changes in the number of biased arguments (p-value = 0.1973) and counterarguments (p-value = 0.8052) can not be considered significant.

7.6.2. Decisions.

Overall we found 641 decisions - 266 biased, 281 non-biased and 94 neutral. 52% of decisions before the workshop were biased, compared to 31% after. Only one had a larger percentage of biased decisions after the workshop. In the case of all the other teams, the percentage of biased decisions decreased. The increase in the number of non-biased decisions (p-value = 0.0024) and the decreased percentage of biased decisions (p-value = 0.0020) were significant. However, the change in the number of biased decisions (p-value = 0.0732) can not be considered significant.

7.6.3. Cognitive biases.

Overall, we found 1110 bias occurrences - 558 before and 552 after the workshop. The sum of these counts is different from the number of arguments since: (a) it was possible for various biases to influence one argument, (b) some biased statements were not connected to any architectural decision.

There was no significant change in the overall number of biases between Task 1 and Task 2 (p-value = 0.8647). This means that the debiasing effect (the smaller percentage of biased decisions and arguments) was not achieved by decreasing the number of bias occurrences during the tasks. In fact, the effectiveness of the debiasing treatment comes from increasing the number of non-biased arguments.

7.6.4. Debiasing techniques.

We compared the amounts of technique uses before and after the workshop, since it was possible for participants to spontaneously use a specific technique during Task 1. We identified 133 uses of the proposed techniques - 26 techniques before and 107 after the workshop.

The number of uses of the practices increased significantly during Task 2 (p-value = 0.0005). However, three teams did not increase their use of the anti-bias techniques substantially after the workshop. Despite this, these teams' percentage of biased arguments and decisions decreased during Task 2. Additionally, two teams, despite using a higher number of debiasing techniques during Task 2, had more biased decisions and more biased arguments during Task 2.

Overall, the anti-optimism technique was used most often (15 before and 57 after workshop), while the anti-anchoring technique was used less often (3 before and 30 after workshop), with the anti-confirmation bias technique rarely being used at all (8 before and 20 after workshop). This may be because listing risks came most naturally, while the other two techniques may require much more effort to be used correctly.

7.7. Discussion

Our results show that the debiasing treatment through the debiasing workshop we designed was successful, both improving the quality of argumentation and design decisions. However, there are some particularities worth discussing in detail, which may help to significantly improve our approach in the future.

Firstly, our approach did not significantly decrease the number of cognitive bias occurrences, biased arguments and biased decisions that impacted our participants (see Section 7.6). Instead, we managed to improve the number of rational arguments and decisions present in the teams' discussions, through which the percentage of biased arguments and decisions decreased. This means that, while it may not be possible to completely get rid of cognitive biases, other ways of rationalising decision-making are possible and could be pursued.

Secondly, the team whose decisions improved the most was the one that had the worst result in Task 1. Furthermore, the team that improved the least was the one with the best result in Task 1. This may mean that, while our treatments successfully improve the performance of initially biased individuals, it may not be as impactful in the case of individuals that were initially less impacted by biases. Since our participants were students with up to three years of professional experience, we do not know yet how different the debiasing effect would be on experienced practitioners (who may be initially less impacted by biases).

Finally, the failure of two teams led us to explore the transcripts in detail. After that, we noticed that their performance dropped at one point, when the participants simply became tired (which they expressed verbally). This is in line with Kahneman's [11] explanation for the existence of two systems – using System 2 is physically exhausting and no human can use it indefinitely. Thus, time for rest may be a crucial factor to bear in mind while attempting debiasing.

7.8. Threats to Validity

Conclusion validity: We used non-parametric tests to examine whether the observed changes in the measured values were significant.

Internal validity: We put significant care into designing the experiment and setting the environment so that no factors other than the workshop influenced the students. The teams did not know the themes for their tasks before the study, and did not have more than 10 minutes of time to interact with the environment outside during the experiment. Finally, to decrease the chances of the researchers distorting the results during the analysis, we used negotiated coding [101] and calculated the results (code counts) for the transcripts only twice.

Construct validity: Our method also improves on the one used in our previous study [159] by taking into account not only arguments and biases but also decisions. This factor is crucial since it is possible that, while the overall argumentation may improve, a team can lack regularity when using rational arguments, thus still making numerous biased decisions nonetheless.

External validity: Our study's weakness is that our participants were all students. While most of them had professional experience, it was limited.

7.9. Conclusion and Future Work

In this paper we explored whether debiasing through a training workshop is an effective method of reducing the impact of cognitive biases on ADM (RQ). We designed such a workshop and examined its effectiveness (Section 7.5). The results show that the debiasing treatment is effective (Section 7.6), although it does not completely eliminate the impact of the biases (Section 7.7).

Through this work, we show that designing a successful debiasing treatment for cognitive biases in ADM is possible, and propose an effective treatment that can become a foundation for future research. Future research can focus on: testing different debiasing techniques, debiasing that takes into account other cognitive biases, exploring the workshop's effectiveness in debiasing experienced practitioners.

Practitioners can use the presented [161] debiasing workshop for training purposes.

8. Debiasing Experts

Article title	Debiasing Architectural Decision-Making: Teaching Software Practitioners.
Authors	**Klara Borowa**, Rodrigo Rebouças de Almeida, and Marion Wiese.
Status	Ongoing review
Contribution	Original research idea, research method design, part of data gathering, most of the data analysis (coding), most of the paper writing.

8.1. Preface

This paper presents a yet unpublished study, which addresses the main limitation of the study from Chapter 7: that it was yet unclear if a similar debiasing effect can be obtained with practitioner participants in the context of real-life projects.

The workshop in this study is of similar design as the one from Chapter 7, with two key differences:

- Since the practice of "monitoring the conversation for confirmation bias" did not result in any visible outcomes, we replaced this debiasing technique. Instead, this study's participants explicitly list architectural alternatives – which is supposed to counter the dangerous combination of anchoring on one solution and then ignoring other possibilities caused by confirmation bias. This technique is also considered a standard architectural decision-making technique [94].
- We added examples illustrating the use of each technique in the workshop.

Overall, 18 practitioners from 3 countries (Brazil, Germany and Poland) took part in the study. They were recruited in pairs, practitioners from one pair had to meet the following criteria:

- They worked on at least one project together;
- They could share details about this system's architecture;
- They worked in similar roles which allowed them to understand and discuss the system's architecture.

One person from each pair participated in the debiasing workshop (workshop group participant), and the other did not (control group participant). Then, both took part in think-aloud sessions where they discussed how an existing architecture could be improved. We compared the argumentation used by the control group and workshop group participants.

Overall, the following positive effects were observed in debiased participants, in comparison to the control group:

- Reduced number of biases,
- Increased number of debiasing techniques,
- Decreased number of biased arguments in favor of particular solutions,
- Increased number of on-biased arguments and counterarguments.

The only unexpected side effect that occurred was a slight increase in biased counterargument count.

Through the analysis of the participant's results, we found that additional factors should be taken into account when attempting to debias practitioners:

- Overuse of low-quality counterarguments,
- Socio-cultural factors –such as respect for leaders, which may have the last word when discussing architectural decisions,
- Practitioner's confidence level – most experienced practitioners are more susceptible to biases,
- Discussing too many architectural decisions in a short time span – this makes practitioners less focused on high-quality argumentation.

For each of these factors, teaching strategies meant to overcome these hurdles are presented in this paper.

8.2. Abstract

Cognitive biases influence decision-making in various areas, including architectural decision-making, where architects face many choices. Prior research suggests that training individuals in debiasing techniques during a practical workshop can help reduce the impact of biases. Our goal was to design and evaluate a debiasing workshop designed for experienced practitioners. To test the workshop's effectiveness, we performed an experiment with 18 practitioners split into control and workshop group pairs. We recorded and analyzed their think-aloud discussions about improving real-world architectures. The workshop successfully reduced the impact of all three researched biases (*anchoring*, *confirmation bias*, and *optimism bias*) and increased the number of arguments and debiasing techniques used by participants. We identified factors that may reduce the effectiveness of debiasing through such workshop: (1) overuse of low-quality counterarguments, (2) socio-cultural factors, (3) confidence level, and (4) discussing too many architectural decisions in a short time span. Overall, we recommend using this workshop as a basis to educate architects and experienced developers, who usually make architectural decisions, about typical harmful influences of cognitive bias on their work and how to avoid them. Finally, we give suggestions on how to improve teaching about architectural decision-making even further, based on this study's findings.

8.3. Introduction

Cognitive biases are systematic errors caused by the heuristics the human mind uses to reduce the complexity of various tasks [160], including decision-making. Cognitive biases distort human decision-making in various domains: from clinicians making erroneous medical diagnoses [165] to software developers copying source code without reading it [20]. In 1995, Stacy and Macmillan [158] first observed the possible impact of cognitive biases in the realm of software engineering. They noticed that *representativeness, availability bias* and *confirmation bias* can influence software developers' daily activities. Since then, research on cognitive biases in software engineering has significantly expanded [14]. The influence of cognitive biases is particularly impactful in the field of architectural decision-making (ADM)[22] [77] since software architecture can be considered as a set of design decisions [6]. However, few studies have focused on behavioral factors (i.e. psychological and human aspects) in ADM, and even fewer have contained any empirical validation of decision-making techniques [83]. Notable example of studies that have focused on improving the process of ADM include those that assess the usefulness of reflection by Razavian et al. [79] (experiment on students), and Tang et al. [157] (experiment on students and professionals).

Debiasing is a process that improves one's judgment by using techniques decreasing the impact of a particular cognitive bias [78]. Teaching software practitioners through a workshop can effectively mitigate some influence of cognitive biases [29]. As far as we know, the only empirically tested debiasing training for ADM was conducted on a group of students with limited experience in software development [166]. This study had students perform theoretical tasks and compared the students' performance based on various different architectures. Given the doubts expressed by the software engineering community about the validity of the findings in experiments with student participants [167], [168], determining whether such training can be effective on experienced practitioners is invaluable. Therefore, the **goal** of this study is to conduct a debiasing intervention with experienced practitioner participants.
As such, we strive to answer the following research question:

RQ: How are experienced practitioners influenced by the proposed architectural decision-making debiasing workshop?

In particular, we explored whether the following two positive effects would appear:

- **Would the workshop decrease the number of cognitive bias occurrences?**
- **Would the workshop increase the participant's use of debiasing techniques?**

To answer this research question, we performed nine controlled experiments with 18 experienced software practitioners, divided into control and training(workshop) groups. Participants were recruited from three countries ([hidden for double-blind review]) and eight companies.
This study's main **contributions** include:

1. The design and empirical evaluation of a workshop that reduces biases impacting practitioners. Our workshop was successful in decreasing the occurrence of all three researched biases (*anchoring*, *confirmation bias*, *and optimism bias*).
2. A workshop that had two statistically significant improvements: (1) increasing the number of non-biased counterarguments used by practitioners, and (2) making the participants list different solution alternatives.
3. We found that the following factors can negatively impact debiasing effects: (1) overuse of low-quality counterarguments, (2) socio-cultural factors, (3) confidence level, and (4) discussing too many architectural decisions in a short time span.

Section provides information on previous research relevant to this study. Section describes the specifics of our research method. The results of the experiment are presented in Section and discussed in Section . Section explores the threats to validity. Finally, we summarize this study in Section .

8.4. Related work

8.4.1. Cognitive biases

The seminal work of Tversky and Kahneman from 1974 [160] introduced the concept of cognitive biases and described three of them: *representatives*, *availability*, and *anchoring*. In their work, the researchers found that human beings rely heavily on heuristics during the decision-making process while often being blind to logical, statistical facts.

These findings later evolved into the dual process theory, describing the human mind as divided into Systems 1 and 2. System 1 performs fast and intuitive decisions that heavily rely on heuristics. Inversely, System 2 performs slow decisions that are logical and rule-based. By using energy mainly for important decisions (System 2), this natural phenomenon allows the human body to save precious energy when making simplified decisions (System 1). However, humans are prone to using energy-saving System 1 for decision-making, even in cases that require rule-based thinking. In that case, cognitive biases might occur [12].

As a counterpoint to biased reasoning, **rational reasoning** can be described based on the research by William James [31] as having two components: (1) perception of a specific piece of **factual information**, and (2) a **logical consequence** of this information [31].

Viewing software architecture as a set of design decisions is a well-established concept [6]. It has resulted in a substantial amount of research regarding issues related to ADM, such as documenting architectural decisions [169], models of architectural decisions [81], decision-making best practices [94], and human aspects of ADM [75], particularly cognitive biases [77].

The impact of cognitive biases on ADM can have severe consequences, such as designing sub-par solutions [170] or incurring dangerous architectural technical debt [171].

Based on previous research on cognitive biases in ADM [26], [77], [171], as well as the most often researched biases in software engineering [14], this study focuses on the following three cognitive biases: *anchoring, confirmation bias, and optimism bias.*

Anchoring bias is the decision-maker's preference for initial information/ideas/solutions (which then become an 'anchor') [160]. Anchoring may, for example, influence developers by

Table 8.1. Participants (W-workshop group / C-Control group)

	No.	Pair No.	Group	Age (years)	IT exp. (years)	Role	Company domain	Company size (employees)
Pilot	1	P1	W	28	7	Developer	Digital payment	21-100
	2	P1	C	28	2	Developer	Digital payment	21-100
	3	P2	W	40	18	Analyst	Digital payment	21 - 100
	4	P2	C	37	5	Product Manager	Digital payment	21 - 100
Main	5	P3	W	52	20	Developer	Media	over 8000
	6	P3	C	59	38	Developer	Media	over 8000
	7	P4	W	42	20	Developer	Marketing Services	101-500
	8	P4	C	40	20	Developer	Marketing Services	101-500
	9	P5	W	42	20	CTO	Finance	101-500
	10	P5	C	52	30	Architect	Finance	101-500
	11	P6	W	39	7	Architect	Signal processing	500-5000
	12	P6	C	59	38	Systems Engineer	Signal processing	500-5000
	13	P7	W	46	20	Architect	Retail	over 5000
	14	P7	C	41	19	Architect	Retail	over 5000
	15	P8	W	29	5	Developer	Education	500-5000
	16	P8	C	30	4	Developer	Education	500-5000
	17	P9	W	39	20	Project Manager	Government	101-500
	18	P9	C	42	19	Systems Development Coordinator	Government	101-500

fixating them on the first numerical estimate that appears during a conversation, which may cause their time estimates to be inaccurate [172].

Confirmation bias is the tendency to purposefully search and interpret information to verify one's beliefs [116]. An example of confirmation bias would be developers' proneness to writing unit tests proving that the software works correctly instead of finding defects [154].

Optimism bias is the overestimation of the probability of positive future outcomes [173]. An example of optimism bias is its impact on project risk management, where individuals ignore potential risks while they focus on the pros of their preferred choices [174].

8.4.2. Debiasing

Debiasing refers to the process of identifying and mitigating cognitive biases that may affect decision-making. Possible types of debiasing treatments can be described using the levels proposed by Fischhoff [78]. A and B-level treatments are usually a lecture or some other method of informing practitioners about cognitive biases (A) and how they may impact practitioners (B). Levels C and D require more organizational resources: a C-level treatment requires giving personalized feedback to each debiased person, and the D-level requires extensive long-term training.

In the field of software engineering, two notable C-level debiasing treatments were reportedly successful. Firstly, rationalizing development time estimates by Shepperd et al. [29], where researchers purposefully anchored software developers on pessimistic estimates to counter over-optimistic predictions. Secondly, in ADM, in their short paper, Borowa et al. [166] reported achieving a debiasing effect on a group of student participants. However, **the debiased students were not impacted by biases less often, but instead used more non-biased arguments in their reasoning**. Additionally, the students performed **purely theoretical tasks** and their performance was compared through them performing **two different tasks before and after a workshop**. However, neither this nor any other recent study ascertained whether the effect on experienced practitioners, and real-life architectures, would be similar. Through our study, we have strived to empirically verify whether a debiasing workshop is effective for experienced practitioners, working on existing architectures.

8.5. Method

In this study, we ran an experiment with practitioners built on top of an existing debiasing workshop structure [166]. However, we adapted the workshop and experiment as explained in the following subsections.

The study was divided into two phases: a pilot phase with two experiments, i.e. two pairs of practitioners from one company and the main experiment with seven pairs of practitioners for different companies. While we present some results from the pilot in this paper, the study plan underwent major changes between these phases, which led us to exclude the pilot's data from quantitative calculations (i.e., code count averages and p-values). The differences between the pilot and the main study are also explained in the respective subsections.

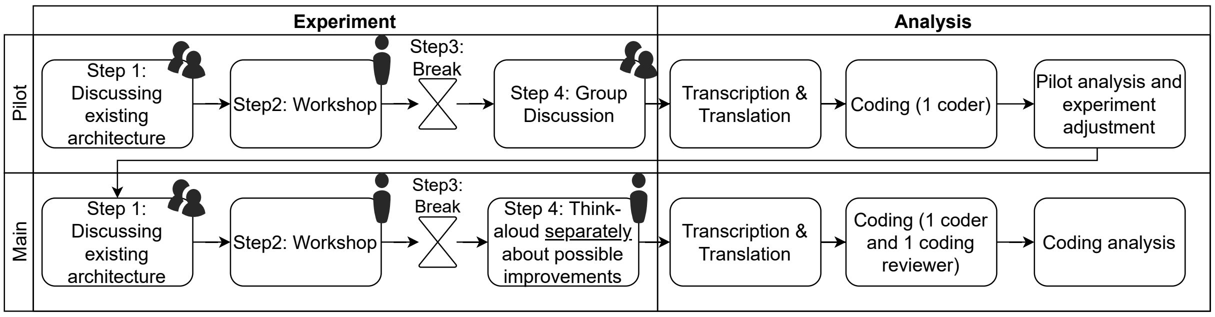
Experiment
Analysis
Pilot
Step 1: Discussing existing architecture
Step2: Workshop
Step3: Break
Step 4: Group Discussion
Transcription & Translation
Coding (1 coder)
Pilot analysis and experiment adjustment
Main
Step 1: Discussing existing architecture
Step2: Workshop
Step3: Break
Step 4: Think-aloud separately about possible improvements
Transcription & Translation
Coding (1 coder and 1 coding reviewer)
Coding analysis

Figure 8.1. Study steps

8.5.1. Sample

We recruited participants through convenience sampling from our network in three countries [hidden due to the double-blind policy]. Convenience sampling was the most appropriate method of acquiring participants for two reasons:

- The experiment required participants to discuss existing systems used by their companies. As such, participants needed to trust the researchers to handle sensitive data appropriately, which allowed us to face real-world designs instead of theoretical examples.
- We needed to find pairs of practitioners who worked together on developing the same system and its architecture and had similar roles in the development team. Explaining the nuances to the participants required personal discussion with one of the researchers.

The participants were informed beforehand that the experiment was related to improving ADM and about the overall steps of the experiment. However, they were not informed that the research was connected with cognitive biases, had no access to the workshop materials beforehand and did not know what tasks they would perform during Step 4 of the experiment. No pair of participants had the chance to contact another pair to disclose such information.

We finished the data gathering for the main experiment after seven participant pairs participated. In the domain of usability testing, where think-aloud protocol studies are widely used, a group of five participants is considered big enough to perform a study [175]. However, we planned to use the non-parametric Wilcoxon Signed Rank Test [176] to calculate p-values of code counts during the analysis. Therefore, we opted for a slightly larger amount of participants since this method is only suited to yield results for at least seven sample pairs. Table 8.1 presents the basic information about the participants, such as experience, role, and domain.

8.5.2. The experiment

In this study, we ran an experiment with practitioners, built on top of an existing debiasing workshop structure [166]. However, we made differences on four specific points.

First, we replaced the least effective debiasing practice, which was that one person monitored a discussion for *confirmation-bias*-influenced statements. Instead, our participants were taught to list multiple solution options for their architectural decisions. Second, we compared the results for the same architectural task done by similar people by comparing pairs of participants, where one attended the workshop (the workshop group) and the other did not (the control group). In contrast, in the experiment of [166], researchers compared varying tasks done before and after the workshop by the same students. Third, our participants were all experienced practitioners, as described in Section . Fourth, the designs discussed in the experiment were real-life cases from the participants' companies.

The experiment itself consisted of the following four steps (see Figure 8.1):

1. *Step 1: Discussing existing architecture (around 30 min.):* Both practitioners took part in a meeting with one researcher. Firstly, they were asked to fill out a simple questionnaire to gather demographic data about themselves (age, experience, role in the company, etc.). Then, they were asked to explain to the researcher the architecture of a system they had both worked on previously. Practitioners had to draw a simple "boxes and arrows" diagram of the architecture, explain the context of the project, and give the researcher the following information: (1) the overall idea behind the system, (2) the system's main components, (3) relationships between components, and (4) the technologies used. This way, we ensured that both practitioners knew the architecture at the same level.
2. *Step 2: Debiasing workshop (around 60 min.):* During this step, the workshop group participant took part in the debiasing workshop, as described in Section . The control group participant had a break.
3. *Step 3: Break:* Research on cognitive biases [12] notes that rational logic-based thinking is physically exhausting. To avoid tiring the workshop group participants more than the control group participants, there was always a break before the last step of the experiment. The time of the break varied but always spanned the time of at least one meal.
4. *Step 4: Architecture improvement (around 30 min.):* Both participants were asked to identify issues with the system's architecture that they presented during Step 1. After that, they were asked to explain what improvements they would propose in order to resolve these problems.

The participants were requested to note the changes on the boxes-and-arrows diagram from Step 1 if appropriate.

In the pilot run with two pairs, we allowed participants to discuss these issues and issues' solutions together. However, in one pilot experiment, we noticed that this approach caused one of the participants to omit their opinions. This could be caused by the fact that the other participant was their superior and dominated the discussion.

In the main experiment, we changed our data gathering approach to a "think-aloud protocol" session [177]. This method was suggested by Razavian et al. [83] for studying behavioral factors in ADM. Accordingly, the participants proposed their improvements separately in the subsequent sessions to avoid interference between them. To understand the participants' thought processes, they were asked to voice their thoughts, i.e., "think aloud", during the whole process. We could not use evaluation methods such as questionnaires or having the participant write down their argumentation since cognitive biases occur during an individual's thought process, and their occurrence may not be visible when the participants have additional time to curate their answers.

The experiment's Step 4 was recorded, with the audio recordings transcribed for further analysis. All the experiment sessions were performed in the native language of the participants by a native-speaking researcher, who subsequently translated the transcript to English to make it accessible to all the authors. We allowed participants to choose between on-site and online participation. Four experiments were performed during a personal meeting (two pilot and two main), and five were held online (all main). The experiment plan is available as part of the additional material [178].

8.5.3. Debiasing Workshop

Our workshop was an intervention that included not only informing about cognitive biases and the specific effect that they may have on the participants but also training the participants through a teaching experience with personalized feedback. Therefore, it is a level C debiasing intervention on Fischoff's debiasing scale [78]. Such an approach proved effective in debiasing software developers to increase the accuracy of effort estimates [29].

Overall, the workshop comprised (1) a short lecture about cognitive biases and the work of Daniel Kahneman and Amos Tversky in general [160] [12], (2) an explanation of how cognitive biases affect ADM in particular, (3) a design session during which the participant was supposed to design a solution for a fictional architecture task and make use of the debiasing techniques. We taught the following debiasing techniques to the participants:

1. *Generating multiple solution options:* This technique aims to counter *anchoring* and *confirmation bias*. This bias combination may lead architects to overlook issues with a solution (*confirmation bias*) after deciding on an initial one (*anchoring*). We propose this technique since it is a basic reasoning technique suggested by many ADM researchers [94] [179] [180].
2. *Listing at least one drawback of the solution alternative:* This technique is targeted towards countering the effects of *anchoring*, since *anchoring* often leads designers to be over-focused on one solution's advantages [166]. (Drawbacks, unlike risks, always have an impact on the solution, e.g. a particular component's license is expensive.)
3. *Listing at least one risk associated with the solution alternative:* Since individuals are prone to over-optimistic predictions, explicitly discussing risks is targeted towards countering *optimism bias* [166]. (Risks, unlike drawbacks, can have an impact on the solution. However, the impact of a risk is determined by its probability of occurrence, which means there is a possibility that the risk may not occur at all.)

The researcher leading the workshop demonstrated the debiasing techniques with an example and actively assisted the participants in the design session when they were struggling with the techniques. This was done to provide personalized feedback required of a level C debasing intervention [78].

8.5.4. Data Analysis

For the data analysis procedure, we used the hypothesis coding technique [164], which involved creating a set of codes before the coding process. The codes that we used are listed in Table 8.2.

Recordings from the experiment sessions were transcribed and translated to English by the researcher organizing the workshop, then coded by another author and, in the main experiment's case, the coding was reviewed by an additional author.

In the pilot phase sessions, the coder was unaware of which transcript belonged to the workshop or control groups. In the main experiment, we took this concept even further and, as suggested by Ericsson and Simon[177], employed **context-free coding** as follows:

1. The workshop organizer divided the transcripts into segments (one for each architectural

decision discussed). This was a natural split since the participants usually discussed one possible change to the architecture after another.

2. For each pair of participants, the control and workshop groups' segments were randomly ordered.
3. Another researcher coded the segments, without knowledge of whether the particular segment belonged to the workshop or control group participant.
4. An additional researcher reviewed the coding and discussed changes with the original coder until a consensus was reached on all codes.
5. The workshop organizer revealed the participants' information to create a summary of each workshop's coding.

We anticipated that a coder might unconsciously try to find more bias-related codes for the control group participants. The context-free coding method aims to decrease the possibility of this researcher's bias affecting the study's outcome.

We evaluated our codes as follows: (1) We created the sum of all codes for a particular measurement (see Table 8.2) for the control and workshop participants. (2) We checked whether the measurements for cognitive biases and bias-impacted statements decreased. (3) We checked whether the measurements of debiasing techniques and non-biased statements increased.

In the work of Borowa et al. [166], the authors assumed that if most arguments used while discussing one decision were non-biased, then the decision was non-biased. In our work, we have not followed this assumption since one argument (both biased and non-biased) can be decisive in leading decision-makers to choose a solution despite other arguments.

In order to determine whether the experiment was successful, we defined a set of research hypotheses (H_R):

- In the case of **values that the workshop strived to decrease** (biased arguments/counterarguments, cognitive biases), the research hypothesis (H_R) was that the measurements with the workshop (workshop group) were smaller than without it (control group). Our null hypothesis (H_0) was that the control group measurements were smaller than or equal to the workshop group measurements.
- The opposite was the case with **values that the workshop tried to increase** (not biased arguments/counterarguments, use of debiasing techniques). Where our research hypothesis (H_R) was that the measurements with the workshop (workshop group) were bigger than

Table 8.2. Coding scheme: adapted from [166]

Code	Code meaning	Description
Arg	**Argument**	A statement in **support** of a possible solution alternative.
Carg	**Counterargument**	A statement in **opposition** to choosing a particular solution alternative.
Anch	**Anchoring**	A statement suggesting that the participant is impacted by **anchoring**.
Conf	**Confirmation bias**	A Statement suggesting that the participant is impacted by a **confirmation bias**.
Opt	**Optimism bias**	A statement suggesting that the participant is impacted by an **optimism bias**.
Ddraw	**Decision's drawback**	Use of the **anti-anchoring technique**, i.e., a statement where the participant discusses a drawback of the solution alternative.
Dmulti	**Decision with multiple alternatives**	Use of the **anti-confirmation bias and anti-anchoring technique** , i.e., a statement where the participant mentions more than one solution alternative.
Drisk	**Decision's risk**	Use of the **anti-optimism bias technique**, i.e. a statement where the participant discusses a risk associated with a solution alternative.

without it (control group). In these cases, the null hypothesis (H_0) was that the control group measurements were bigger (or equal to) the workshop group measurements.

In order to examine which results are statistically significant, and as such can be considered as accepted with high confidence level, we performed the non-parametric Wilcoxon Signed Rank test [176] for each of the measurements and the percentage of biased statements (i.e. arguments and counterarguments). This parametric test is used to measure whether a statistical difference can be observed between small paired (dependent) data samples. We consider the control group and workshop group participants to be dependent since they both shared an understanding of the particular architecture and performed the task of discussing improvements for the same systems.

We accept H_R as proven in a statistically significant manner in cases where the p-value is smaller than 0.05, i.e., there is a less than 5% chance that H_R is false. In the case of p-values smaller than 0.1 but greater than 0.05, we cannot accept H_R as statistically significant, but we consider the possibility that this may be due to our limited sample. This means there is a considerable probability that further research might still show the respective research hypotheses can be accepted.

8.6. Results

In this section, we present the results of our analysis. Due to the change in data-gathering methods, the results from the pilot study (Pairs P1 and P2) were not included in any calculations

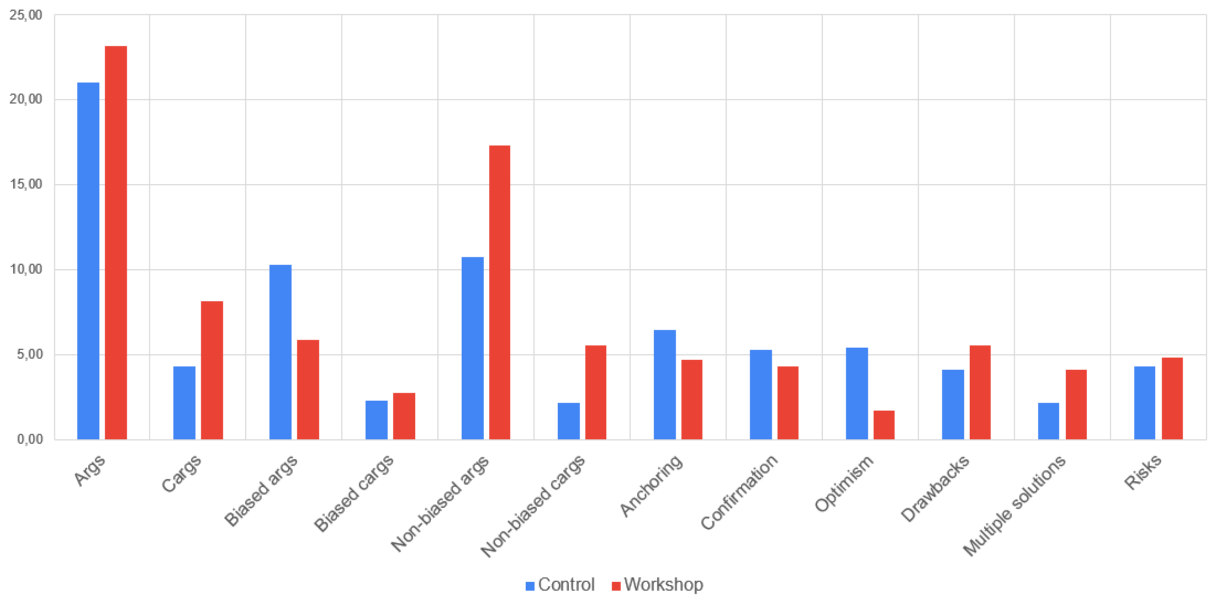

Figure 8.2. Average code sums for each measured value

(p-values, overall code sum/average for control/workshop group) but are presented in this section as well.

Figure 8.2 showcases the average code counts for each measurement totaled across all participants. Overall, when only taking the code averages into consideration, most measured values changed in accordance to the research hypothesis H_R. That is: (1) each of the three researched biases occurrences decreased, (2) the use of all debiasing techniques increased, (3) the amount of arguments and counterarguments increased, (4) the amount of biased arguments decreased.

However, one single measurement was not as we hypothesized. The amount of biased counterarguments slightly increased (from an average of 2.29 in the control group to 2.71 in the workshop group).

8.6.1. Statistical significance

The p-values calculated using the Wilcoxon Signed Rank test [176] are presented in Table 8.3. We calculated the p-values for all measurements that we coded and the percentage of biased statements. Of all measured values, changes were statistically relevant in two cases: (1) The increase of non-biased counterarguments and (2) the increased use of the "listing multiple solutions" debiasing technique.

Table 8.3. p-values for each measurement

Measurement	Control group average of code sums	Workshop group average of code sums	p-value	Research hypothesis (H_R)
Arg	21	23.14	0.3997	Workshop >Control
Carg	4.29	8.14	0.0574	Workshop >Control
Anch	6.43	4.71	0.1721	Workshop <Control
Conf	5.29	4.29	0.4063	Workshop <Control
Opt	5.43	1.71	0.0862	Workshop <Control
Biases sum	17.15	10.71	0.1094	Workshop <Control
Biased Args	10.29	5.86	0.0739	Workshop <Control
Biased Cargs	2.29	2.71	0.4461	Workshop <Control
Not biased Args	10.71	17.29	0.1342	Workshop >Control
Not biased Cargs	2.14	5.57	**0.0449**	Workshop >Control
% Biased statements	49,18%	28,09%	0.0781	Workshop <Control
Ddraw	4.14	5.57	0.1114	Workshop >Control
Dmulti	2.14	4.14	**0.029**	Workshop >Control
Drisk	4.29	4.86	0.5	Workshop >Control
Techniques use sum	10.57	14.57	0.07813	Workshop >Control

Table 8.4. (Non-)Biased Arguments and Counterarguments(B – Biased, NB – Non-biased)

		Control				Workshop				**Control**	**Workshop**	**Difference**
		Arguments		Counterarg.		Arguments		Counterarg.		**Biased**	**Biased**	**Biased**
	Pair	B	NB	B	NB	B	NB	B	NB	**statements [%]**	**statements [%]**	**statements [pp]**
Pilot	P1	5	2	1	2	4	0	1	6	60%	38%	-22
	P2	8	1	0	1	1	3	0	1	80%	17%	-63
Main	P3	12	10	4	1	8	8	2	9	59%	36%	-24
	P4	3	6	1	1	11	5	9	4	36%	67%	30
	P5	14	22	1	4	3	20	0	1	37%	11%	-26
	P6	9	10	2	3	5	13	3	7	46%	26%	-20
	P7	24	17	6	4	10	54	3	13	59%	15%	-43
	P8	3	1	0	1	1	10	0	1	60%	8%	-52
	P9	7	9	2	1	3	11	2	4	47%	24%	-24

8.6.2. Arguments and Counterarguments

Table 8.4 presents the numbers of arguments and counterarguments for each pair of participants. When combined with Figure 8.2, which shows the average of each type of argument in the control and workshop groups, it allows us to make the following observations:

- The number of arguments in support of a solution was greater in the case of the workshop group.
- The workshop group managed to discuss more counterarguments overall.
- Workshop participants used less biased arguments.
- Both the number of **biased and non-biased counterarguments** was larger for workshop participants.

Additionally, based on the data shown in in Table 8.3 we have more observations regarding the statistical significance of the results related to arguments and counterarguments. Unlike Borowa et al. [166], we did not find a statistically significant decrease in the percentage of biased statements or the a statistically significant increase in the number of non-biased arguments. However, this may be due to our limited sample. Since only in one case did the workshop participant exhibit a higher percentage of biased statements than the control group participant (pair P4) such visible effect did occur despite not being statistically significant. However, **the increase of non-biased counterarguments was significant** (p-value = 0.0449). An interesting point is that **the number of biased arguments also decreased notably**. While this decrease is not statistically significant to the 5% significance level, this may be due to our limited sample (p-value = 0.0739).

Table 8.5. Occurrences of Cognitive Bias (ANCH – Anchoring, OPT – Optimism bias, CONF – Confirmation bias)

		Control			Workshop			**Control**	**Workshop**	**Difference**
	Pair	ANCH	CONF	OPT	ANCH	CONF	OPT	**Bias sum**	**Bias sum**	**Bias sum**
Pilot	P1	6	3	1	1	1	1	10	3	-7
	P2	7	3	1	1	0	0	11	1	-10
Main	P3	9	7	2	7	4	0	18	11	-7
	P4	3	1	2	12	11	0	6	23	17
	P5	10	3	8	1	2	3	21	6	-15
	P6	9	3	0	5	3	1	12	9	-3
	P7	6	21	23	6	6	4	50	16	-34
	P8	1	0	3	0	0	2	4	2	-2
	P9	7	2	0	2	4	2	9	8	-1

8.6.3. Cognitive biases

While Figure 8.2 shows the overall sum of bias occurrences in each of the control and workshop groups, Table 8.5 presents the number of bias occurrences from each participant pair. Overall, the workshop group participants were less impacted by each of the researched biases, with only one workshop participant being more susceptible to biases than their colleague (pair P4). While these overall decreases can not be considered statistically significant to the 5% level, they are noticeable in almost every participant pair for every researched bias. The one sole exception to this is pair P4.

In Borowa et al.'s research [166], the student participants of the workshop did not exhibit decreased susceptibility to cognitive biases at all. This means that in this study, such effect was achieved for the first time.

8.6.4. Debiasing techniques

Table 8.6 shows how many debiasing techniques were used by each participant, while Figure 8.2 shows the sums of each technique's use in the workshop and control groups.

Overall, the workshop participants used each of the three techniques more frequently than the control group participants; this was not the case only for pair P5. Additionally, the **"listing multiple options" technique was used significantly more (p-value = 0.029)** by workshop participants.

8.6.5. Decisions

Table 8.7 showcases how many architectural decisions were discussed by each participant. We did not attempt to influence the decision count through this experiment, so no p-values were calculated. However, it may be notable that in most cases(six out of nine), the workshop participants discussed fewer architectural decisions than their control group colleagues.

Table 8.6. Use of Debiasing techniques
(DRAW – Stating drawback, MULTI – Listing solution alternatives , RISK – Stating risks)

		Control			Workshop			**Control**	**Workshop**	**Difference**
	Pair	DRAW	MULTI	RISK	DRAW	MULTI	RISK	**Techniques**	**Techniques**	**Techniques**
Pilot	P1	2	0	2	6	1	2	4	9	5
	P2	1	0	0	1	0	1	1	2	1
Main	P3	4	0	2	5	3	2	6	10	4
	P4	2	0	2	2	5	0	4	7	3
	P5	8	5	11	5	5	7	24	17	-7
	P6	0	1	0	5	3	2	1	10	9
	P7	9	9	12	13	9	20	30	42	12
	P8	2	0	1	5	1	2	3	8	5
	P9	4	0	2	4	3	1	6	8	2

Table 8.7. Decisions discussed

	Pair number								
	P1	P2	P3	P4	P5	P6	P7	P8	P9
Workshop	4	4	6	9	11	11	9	5	16
Control	5	2	11	19	8	12	6	12	18
Difference	-1	2	-5	-10	3	-1	3	-7	-2

8.7. Discussion

Our debiasing workshop **succeeded in decreasing the occurrence of all three biases** (*anchoring*, *confirmation bias*, *optimism bias*) for the workshop participants. Although this decreases were not statistically significant, this had never been achieved previously.

The amounts of uses of **each debiasing technique that we taught increased** for workshop participants as well.

Additionally, we observed **two statistically significant changes** in the case of the workshop group: (1) the increase of non-biased counterarguments and (2) the increased use of the debiasing technique of "listing multiple solution options".

While most measured values did change as we hypothesized, in most cases, the results were not statistically significant. However, this does not necessarily mean this study's results are not valuable. It does mean however, that not statistically significant results have weaker evidence supporting their validity and should be considered more carefully. Proper calculation of statistical significance in the case of such small samples is not easily achievable, and we used the p-values in this study mainly to distinguish results with the highest validity.

We found it intriguing that, in comparison to the study on students [166], the frequency of non-biased arguments did not increase in a statistically significant manner. This implies that, for practitioners, the increase in non-biased counterarguments was more severe than in non-biased arguments. We suspect this may be due to the practitioners' having the preexisting skill of arguing in favor of their chosen solutions since they most likely had to do this repeatedly in their workplace. On the other hand, students may find the formulation of both pro and con arguments new since university courses do not usually allow students much creative freedom with architectural decisions.

8.7.1. Lessons learned

Some of the study's participants produced notably **unexpected results.**

A side effect of the workshop seems to be a slight increase in the use of biased counter-arguments by workshop participants, most likely because the workshop focused on negative factors such as decision-related risks and drawbacks. As such, workshop participants may have over-focused on creating numerous counterarguments with less care about their quality. However, the beneficial increase in non-biased counterarguments was significantly higher.

Teaching suggestion 1: Focus on high-quality counterarguments. Compared to students, experienced practitioners have fewer problems in specifying non-biased arguments in support of decisions. As such, when teaching practitioners, more focus should be on discussing fact-based counterarguments.

Additionally, Participant No 3 (Pair P2 from the pilot) discussed only two design decisions, which was the smallest number of decisions discussed by all the participants. In this case, we suspect that this resulted from **a socio-cultural factor** associated with group decision-making [181] - the junior participant felt internal pressure due to the presence of the senior participant (control group).

Teaching suggestion 2: Debiasing might only be effective when team members of all seniority levels take part in it. Practitioners must be made aware that this training's debiasing effect can be significantly decreased if thr participant's colleagues who are higher in the hierarchy dismiss their concerns. As such, teaching only younger developers while omitting senior team members or leaders may not be effective.

Furthermore, the use of debiasing techniques also increased for almost all participant pairs, except one. In the case of pair P5, Participant No 9, who was the workshop group participant, was also the CTO of the company. During the experiment, the author conducting it noticed that the attitude of the participant suggested that they considered the contents of the workshop to be trivial. The participant's **confidence level was most likely heightened due to their high position in the company**, where they were previously the head architect for many years. It is likely that this made them more susceptible to cognitive biases and less likely to use the debiasing techniques.

Teaching suggestion 3: Focus on high-level and experienced team members. If the participant has a high position in the company and vast engineering expertise, they may have a high confidence level in the architecture of their system and thus be less likely to learn from this workshop. In the case of these participants, take care to specifically inform them of this danger.

Finally, participant No. 7 (Pair 4) was the only workshop group participant who gave more biased statements than the control group participant. The only metric that seemingly distinguishes participant No. 7 is that they choose to discuss **the highest number of architectural decisions (19)** from all the participants. This is in line with classic research on cognitive biases, which defines them as a natural heuristic used by the human mind to avoid overload with too many tasks [12]. This means that when too many decisions are considered, the human mind will behave in an energy-saving manner that makes biases more likely to occur.

Teaching suggestion 4: Avoid discussing too many decisions. Inform participants that considering too many architectural decisions in a short time span is likely to produce an increase in cognitive bias influences. Try to moderate the training in such a way that focuses on high-quality argumentation of a lower number of key decisions.

8.8. Threats to validity

Our threats to validity are described based on the guidelines for experiments in software engineering created by Wohlin et al. [182]:

Construct Validity As we performed a controlled experiment, our study heavily relies on its construct validity. While we carefully designed the experiment to maximize construct validity (see Section), one threat that could impact the results is how each researcher could lead the debiasing workshop differently. To handle the potential particularities and cultural factors that could affect the experiment, we discussed possible scenarios that could occur during the experiment before conducting it and then used the same workshop plan and slides. Furthermore, we performed a pilot run of the workshop to enhance our study's design. When we observed that one pilot study participant being the other's superior may be such an additional factor impacting validity, we changed the last step of the experiment from group discussion to separate think-aloud protocol sessions. Additionally, we had the participants take a break before Step 4 of the study, to avoid overtiring workshop participants. Finally, every control/workshop pair of participants discussed the same architecture during Step 4 of the study, which makes the results for both groups simpler to compare. This is a difference from previous research that performed the comparison of before&after the workshop, using two different architectural tasks [166].

Internal Validity Our experimental environment was carefully controlled to ascertain internal validity. We took care to not allow external factors to impact the participants: (1) We did not tell them about cognitive biases beforehand. (2) They did not know about Step 4 of the experiment beforehand, so they could not prepare for it. (3) During Step 4, participants did not get the chance to contact anyone or use the internet for help. Additionally, we employed context-free coding to avoid the coder's unconscious desire to showcase that the workshop works as planned. An additional author then reviewed this coding.

Conclusion Validity Regarding conclusion validity, we have the threat of a low number of experiments executed, which is due to the fact that it is hard to find practitioners who are willing to invest three hours into a scientific experiment. Finding participants was particularly challenging since our experiment required (1) pairs of participants who worked together on the same project and (2) sensitive data about their real-life architecture. However, this low sample size puts a risk on the statistical significance of the findings. To counteract this threat, we performed the non-parametric Wilcoxon Signed Rank Test, which makes it possible to check the statistical significance of small paired samples. Still, a larger sample would provide more confidence in the findings.

External Validity As common for a controlled experiment, the external validity is hampered by the strictly controlled setting. However, we did try to perform the experiment in the participants' preferred environment: in their company's office or using their preferred videoconferencing software. Additionally, all our participants were experienced practitioners who discussed real-life architectures from various domains.

8.9. Conclusion

We performed a debiasing workshop in order to explore its impact on experienced software practitioners. The design of our study was based on the structure by Borowa et al. [166], with some adaptations (as explained in Section). We divided participants into separate treatment (workshop) and control groups as it is common in multiple studies [75], [79]. We also employed the think-aloud protocol study method, as suggested by Razavian et al. [83] as appropriate for researching behavioral aspects of ADM.

The research question that we aimed to answer was:

RQ: How are experienced practitioners influenced by the proposed architectural decision-making debiasing workshop?

Through this study, we found that:

1. Our workshop successfully decreased occurences if cognitive bias for all three researched biases (anchoring, confirmation bias, and optimism bias).
2. The use of the proposed debasing techniques that were taught increased.
3. Two effects were statistically significant: (1) the rise of non-biased counterarguments and (2) the increased frequency of using the listing multiple solutions technique.
4. There are additional factors impacting successful debiasing ADM, such as: (1) Overuse of low-quality counterarguments, (2) socio-cultural factors, (3) confidence level, and (4) discussing too many architectural decisions in a short time span.

Educators intending to perform such a debiasing workshop in practice can use the teaching materials we prepared for this study either directly or using them as a basis for their own materials [178]. Additionally, we encourage them to employ the suggestions presented in Section .

For future work, we propose focusing on longitudinal studies in a real-world setting to ensure that the debiasing treatment has a long-term impact.

8.10. Data availability

The experiment plan, workshop plan, and workshop slides are available on Zenodo [178].

9. Discussion and limitations

This thesis is based on a series of research papers (Chapters 3–8), which focused on the following aspects of architectural decision-making: architectural decision rationales, cognitive biases impact on architectural decisions, the impact of cognitive biases on architectural technical debt and the design of a debiasing intervention meant to alleviate the negative impact of cognitive biases on architectural decisions. The results of these studies are discussed below.

Previously, empirical evidence of design rationales in architectural decision-making was limited [83] and based mainly on inflexible surveys asking about predefined rationales [82] or interviews with a small number of participants [85] [84]. The study presented in Chapter 3 employed a mixed-methods approach that allowed for a bigger sample from questionnaires and flexibility (the questionnaires contained open-ended questions and were supplemented by interviews). Additionally, data analysis considered the practitioner's experience level, which led to new findings. As such, it is the most mature work on architectural design rationales so far.

The research revealed that practitioners' architectural decisions were mainly motivated by the following rationales: ease of use for development activities, maintainability, performance, prior knowledge/experience in using the solution, and time/deadline. These results partially confirm the findings of Weinreich et al. [84], who discovered that the most impactful rationale was "Personal experience / Preferences." Additionally, the quality attributes of portability and compatibility were found to be of very low significance to practitioners since modern technologies (such as containerization) and communication standards (such as REST API) were considered a universal answer to issues with portability and compatibility.

Finally, practitioners in the middle of their careers (5 to 14 years of experience) were the only experience group that preferred choosing architectural solutions they had no prior experience with to try something new. This effect could be attributed to their confidence (unlike less experienced practitioners) and thirst to improve their skills to obtain a promotion (unlike most experienced practitioners).

The most relevant rationales that motivated practitioners' architectural decisions were not solely based on facts about the software's quality or the limitations of the project. Although practitioners did consider them (mainly maintainability, performance, and deadlines), the decisions were based mostly on personal preferences, experiences, and the hope to decrease their own workload. **These three, however, are all possible cognitive bias antecedents**. Preferences are naturally based on a small sample of experiences, and cognitive biases (such as representativeness [33]) are often the result of erroneous perceptions of reality based on an individual's experiences [11].

Contribution 1: Set of rationales which motivate architectural decisions made by practitioners.
Many of these rationales seem to be cognitive bias antecedents. In particular, the practitioner's personal preferences, experiences, and hope for a decreased workload.

While researchers have been interested in the impact of cognitive biases on architectural decision-making for over a decade [22], they usually investigated a small number of pre-selected biases [77]. This means that no researcher before has explored a broader scope of possible bias occurrences in architectural decision-making without making prior assumptions about which biases whould be relevant.

Chapter 4 presents a study that broadly assesses cognitive biases' impact on architectural decision-making and their consequences. The novelty of this study stems from its data-gathering method. Instead of pre-selecting bias definitions before data gathering, a more flexible approach was taken. Participants learned the definition of cognitive biases and then described real-life situations in which they believed a cognitive bias occurred. During the data analysis, these descriptions were mapped to the definition of particular cognitive biases. The cognitive biases that were shown to influence architectural decision-making are: the framing effect, confirmation bias, the IKEA effect, Parkinson's law of triviality, anchoring, the curse of knowledge, pro-innovation bias, the planning fallacy, the bandwagon effect, irrational escalation, the law of instrument, and optimism bias. This list partially confirms the findings of van Vliet and Tang [19], who found that anchoring, the framing effect, and confirmation bias impacted architects.

Chapter 4 additionally presents specific situations when these biases may occur. For example, the following three types of situations were rated as most often occurring by the study participants:

- Assessment of a system based on marketing information (framing effect).
- Judging the complexity of a problem based on the client's expectations instead of factual data (confirmation bias).
- Overvaluation of the quality of a system by its creators (IKEA effect).

Overall, the list of biases developed in this study is a foundation for future research on cognitive biases in architectural decision-making. It enables researchers to focus on the biases most likely relevant to architectural decision-making instead of exploring the space of over 37 cognitive biases detected in various software engineering papers [14].

Furthermore, this study defines which aspects of architectural decision-making can be impacted by these biases. These include mainly: the architect's preferences, the scope of considered alternatives, and the perception of requirements. The impact on the architect's preferences is of particular importance since this exact aspect matches the most influential rationale found in chapter 3 and previous research [84].

Contribution 2: Set of cognitive biases that impact architectural decision-making. This list includes which aspects of architectural decision-making may be influenced by each of these biases.

Only a small number of researchers have suggested previously that cognitive biases may influence the occurrence of technical debt. However, they did not provide any empirical evidence for such a relationship [20] [128] [127]. Additionally, it has been proven that architectural decisions may lead to incurring architectural technical debt [25]. As such, it seemed likely that cognitive biases might make practitioners more likely to incur architectural technical debt. Since architectural technical debt, in particular, is considered the most dangerous type of technical debt [24], it was crucial to resolve whether this impact actually existed.

Research presented in Chapter 5 proves that cognitive biases may make software architects incur unnecessary technical debt. Almost all biases listed in Chapter 4 were possible culprits, with three being the most influential: anchoring, confirmation bias, and optimism bias. Notably, anchoring and optimism bias impacted the occurrence of "Architectural Lock-in" technical debt items [127]. Incurring of this type of debt can result from a specific bias interaction: first, practitioners choose the first solution that they can find (anchoring) and then take a "leap of faith" in the hope that it will be satisfactory without considering the risks of their decision (optimism bias). Another notable architectural technical debt item was "Re-inventing the wheel," [127] which may be caused by confirmation bias, when the practitioners unnecessarily write major components from scratch instead of searching for high-quality, pre-existing solutions.

Contribution 3: Cognitive biases may make software architects incur unnecessary architectural technical debt.

Three cognitive biases are the biggest culprits with regard to technical debt: anchoring, confirmation bias, and optimism bias.

Cognitive biases may impede the architectural decision-making process, significantly decreasing the software's quality. Incurring unnecessary architectural technical debt is only one example of such impact. Cognitive biases, in general, disrupt basic decision-making aspects, such as searching for solution alternatives, and the perception of requirements can be impacted by cognitive biases (see Chapter 4).

To minimize this effect, debiasing techniques are needed. However, this is a major challenge since there is clearly a lack of empirically verified debiasing strategies in software engineering [14].

One such study was done by Shepperd et al., who focused on improving development time estimates [29]. While empirical evaluation studies exist on improving architectural decision-making [79] [157], none before specifically targeted cognitive bias prevention. This has been addressed in this thesis, by the development and evaluation of a successful debiasing intervention (Chapters 6-8).

Chapter 6 showcases a pilot study that tested whether a B-level [78] intervention (informing about biases through a presentation) would be effective. While it did not provide a debiasing effect, it showcased how the three researched biases (anchoring, optimism bias, and confirmation bias) interacted with each other, strengthening their negative impact.

When making architectural decisions, architects often **anchor** on the first solution they came up with. Then, opposition against such decisions is silenced by **confirmation bias** – often, the symptom of this is the "We already decided on that" argument used by coworkers. Finally, **optimism bias** affects the overall atmosphere during decision-making, making practitioners believe that "everything will be ok" while ignoring possible risks.

This triad of biases is most likely fueled by the rationales found in Chapter 3. Practitioners prefer solutions that minimize the required effort for development overall. However, information about what is "simple and easy" comes from their prior experiences, which may not be reliable.

💡 **Contribution 4: The wicked triad – anchoring, optimism, and confirmation bias.**

These three cognitive biases combine to cause problematic architectural decisions: fast anchoring on a solution, optimistic belief that no more consideration is necessary, and avoidance of any information that does not fit this narrative (confirmation bias).

Based on these three observed bias impacts, three debiasing techniques meant to counter them were proposed:

1. Discussing solution drawbacks;
2. Discussing solution risk;
3. Having someone monitor the discussion for the "but we already decided on that" argument.

Chapter 7 showcases the results of a C-level [78] debiasing workshop, during which students were taught about cognitive biases in architectural decision-making, and they participated in practical exercises for using the debiasing techniques. This workshop successfully increased the student's use of non-biased (i.e., based on factual data) argumentation for the discussed architectural solutions. However, the number of bias occurrences did not change noticeably. Additionally, one of the three debiasing techniques was seldom used by participants during group discussion, that is, having someone monitor the discussion to avoid the "but we already decided on that" argument.

Subsequently, the workshop was further improved: the "Have someone monitor the discussion" debiasing technique was replaced with the "listing multiple solutions" technique. Then, the workshop was empirically verified on 18 experienced practitioners from three countries (Chapter 8). In this case, the workshop improved almost all measured values:

- The number of non-biased (i.e., based on facts) arguments increased,
- The number of non-biased counterarguments increased,
- The number of biased arguments decreased,
- The number of anchoring occurrences decreased,
- The number of confirmation bias occurrences decreased,
- The number of optimism bias occurrences decreased,
- The number of uses for each of the three debiasing techniques increased.

Contribution 5: Empirically validated debiasing workshop.
The debiasing workshop was proven to be an effective debiasing intervention for both students and experienced practitioners.

Another finding of this thesis is that the debiasing effect on students and practitioners seems slightly different.

The workshop made **students** (see Chapter 7) use more non-biased arguments and counterarguments, but it did not decrease the number of bias occurrences. In the case of **practitioners** (see Chapter 8), the number of bias occurrences impacting workshop participants was actually smaller. The workshop also had a beneficial effect in decreasing the number of biased arguments and increasing the number of non-biased arguments and counterarguments. However, practitioners also exhibited a slight increase in their biased counterarguments count, which was a negative side effect of the debasing workshop. This leads to the conclusion that while beginners have to be motivated to argue their choices, experts have to be trained on the quality of the argumentation. Still, in both groups, using counterarguments (and, as such - considering drawbacks and risks) seems to be the key to rational decision-making.

Contribution 6: Students and experienced practitioners react differently to a debiasing workshop.

Still, in both cases, the key seems to lay in training focused on high-quality counterarguments, which consider architectural solutions' drawbacks and risks.

9.0.1. Limitations

As in the case of any research, its **limitations** have to be considered. Firstly, Chapters 3–5 are based on reports on practitioners' previous experiences. It is possible that their memory of past events was not fully accurate or that they based their knowledge on assumptions, e.g., about their colleague's decision-making, was faulty. Participants may have also decided not to share the whole truth of their experiences, for example, due to shame about their own past mistakes. In all of these cases, we attempted to counter this by finding a diverse sample of practitioners.

Additionally, a limitation that has to be considered is that the debiasing interventions on students and practitioners differed in both data gathering and data analysis methods. As such, comparing the results from these studies may lead to unintentionally erroneous assumptions. The last experiment on practitioners can be considered the most mature of all the debiasing attempts since it is based on a comparison of practitioners performing the same architectural tasks, and the data analysis includes context-free coding [177], which made the coding researcher unable to determine whether they are coding the control or workshop participant's transcript segment.

10. Conclusion

This thesis presents a multi-step, in-depth study of cognitive biases in architectural decision-making. Five of the six papers presented in this thesis have already been published in peer-reviewed venues. These make it possible to answer the following research questions:

RQ1: What rationales are the main reasons behind decisions impacting software practitioner's architectural decision-making?

This work showcases a list of common rationales behind architectural decisions (Table 3.3). The most common rationales with which practitioners justify their decisions include ease of use for development, maintainability, performance, prior knowledge/experience in using the solution, and time/deadlines. Additionally, the origins of why participants used these rationales are presented as well (Section 3.6.2). The most prominent rationale's origins include: practitioner's prior experience, client focus, decreased workload.

RQ2: How do cognitive biases impact architectural decision-making?

This thesis presents a list of 11 cognitive biases that affect architectural decision-making (Section 4.5.2), as well as a detailed description of the specific impacts of these biases.

RQ3: Does cognitive biases' impact on architectural decision-making cause architectural technical debt? The work presented in this thesis reports how cognitive biases may impact decisions related to architectural technical debt. The specific biases that are the most prominent in this case are optimism bias, confirmation bias, and anchoring (Table 5.4). Additionally, the antecedents that preceded the occurrence of these biases are discussed(Section 5.6.4), as well as the types of architectural technical debt incurred due to biases (Section 5.6.3).

RQ4: How can the negative impact of cognitive biases on architectural decision-making be alleviated? It is possible to alleviate the impact of cognitive biases on architectural decision-making. However, successful debiasing requires at least C-level debiasing interventions [78], i.e., simply educating about cognitive biases and their impact on architectural decision-making is not enough. Debiased persons have to learn debiasing techniques through practical exercises.

This thesis shows three experiments that resulted in a successful debiasing intervention:

1. A B-level intervention (a lecture without practical exercises nor information on debiasing techniques) on a small group of students: there was no debiasing effect, but the wicked bias triad was identified, and a set of debiasing techniques was proposed.
2. An experiment on 12 groups of students that took part in a C-level debiasing workshop,

which resulted in the increased use of rational arguments and counterarguments. However, the bias occurrences did not decrease.

3. An improved C-level debiasing workshop was tested with 18 practitioner participants who discussed real-life architectural designs. **In their case, a decrease in bias occurrence was observed**.

.

This thesis, overall, presents a body of work that significantly expands existing knowledge on cognitive biases in architectural decision-making with:

- The understanding of the impact of cognitive biases on architectural decision-making, with a particular focus on architectural technical debt.
- A successful, empirically validated, debiasing intervention.

Future work could focus on the longitudinal effects of the debiasing interventions since the debiasing effect may possibly diminish over time. Additionally, such techniques as shadowing could be used to observe practitioners in their everyday work and to observe the changes in practitioners' behavior during real-life architectural decision-making. Furthermore, besides the wicked triad, other bias interactions may exist. As such, future research on bias interactions in architectural decision-making may be appropriate.

10.0.1. Data availability

Data for the debiasing workshops from Chapters 7 and 8 are available online [161] [178]. As such, they may be used by anyone wishing to perform a debiasing intervention.

References

[1] Boehm, “Software engineering”, *IEEE Transactions on Computers*, vol. C-25, no. 12, pp. 1226–1241, 1976. DOI: `10.1109/TC.1976.1674590`.

[2] P. Naur and B. Randell, *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*. 1969.

[3] P. Bourque, R. E. Fairley, and I. C. Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd. Washington, DC, USA: IEEE Computer Society Press, 2014, ISBN: 0769551661.

[4] J. Buxton and B. Randell, *Software Engineering Techniques: Report on a conference sponsored by the Science Committee, Rome, Italy, 27th to 31st October 1969*. 1960.

[5] D. E. Perry and A. L. Wolf, “Foundations for the study of software architecture”, *ACM SIGSOFT Software engineering notes*, vol. 17, no. 4, pp. 40–52, 1992.

[6] A. Jansen and J. Bosch, “Software architecture as a set of architectural design decisions”, in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA’05)*, IEEE, 2005, pp. 109–120.

[7] P. B. Kruchten, “The 4+ 1 view model of architecture”, *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

[8] ISO/IEC/IEEE, “Systems and software engineering – architecture description”, *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pp. 1–46, Jan. 2011.

[9] S. Brown, “The c4 model for software architecture”, *Updated August*, vol. 1, 2018.

[10] A. Tversky and D. Kahneman, *Judgment under uncertainty: Heuristics and biases. utility, probability, and human decision making, 185 (4157), 141–162*, 1975.

[11] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[12] D. Kahneman, “Think fast, think slow”, *Farrar, Straus and Giroux, New York*, 2011.

[13] NobelPrize.org, *Daniel kahneman – facts*, `https://www.nobelprize.org/prizes/economic-sciences/2002/kahneman/facts/`, Accessed: 03.01.2024.

[14] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, “Cognitive Biases in Software Engineering: A Systematic Mapping Study”, *IEEE Transactions on Software Engineering*, vol. 5589, no. c, 2018, ISSN: 19393520. DOI: `10.1109/TSE.2018.2877759`.

[15] R. Mohanani, P. Ralph, and B. Shreeve, “Requirements fixation”, *Proceedings - International Conference on Software Engineering*, pp. 895–906, 2014, ISSN: 02705257. DOI: 10.1145/2568225.2568235.

[16] T. Halkjelsvik and M. Jørgensen, *Time Predictions: Understanding and Avoiding Unrealism in Project Planning and Everyday Life*. Springer Nature, 2018.

[17] K. Borowa, S. Kamoda, P. Ogrodnik, and A. Zalewski, “Fixations in agile software development teams”, *Foundations of Computing and Decision Sciences*, vol. 48, no. 1, pp. 3–18, 2023.

[18] I. Salman, P. Rodriguez, B. Turhan, A. Tosun, and A. Güreller, “What leads to a confirmatory or disconfirmatory behavior of software testers?”, *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1351–1368, 2020.

[19] H. Van Vliet and A. Tang, “Decision making in software architecture”, *Journal of Systems and Software*, vol. 117, pp. 638–644, 2016.

[20] S. Chattopadhyay, N. Nelson, A. Au, *et al.*, “A Tale from the Trenches : Cognitive Biases and Software Development”, in *International Conference on Software Engineering (ICSE)*, 2020, pp. 654–665, ISBN: 9781450371216. DOI: 10.1145/3377811.3380330. [Online]. Available: https://doi.org/10.1145/3377811.3380330.

[21] M. Razavian, B. Paech, and A. Tang, “Empirical research for software architecture decision making: An analysis”, *Journal of Systems and Software*, vol. 149, pp. 360–381, 2019.

[22] A. Tang, “Software designers, are you biased?”, *Proceedings - International Conference on Software Engineering*, no. January 2011, pp. 1–8, 2011, ISSN: 02705257. DOI: 10.1145/1988676.1988678.

[23] A. Manjunath, M. Bhat, K. Shumaiev, A. Biesdorf, and F. Matthes, “Decision Making and Cognitive Biases in Designing Software Architectures”, *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, pp. 52–55, 2018. DOI: 10.1109/ICSA-C.2018.00022.

[24] A. Martini and J. Bosch, “The danger of architectural technical debt: Contagious debt and vicious circles”, in *2015 12th Working IEEE/IFIP Conference on Software Architecture*, IEEE, 2015, pp. 1–10.

[25] M. Soliman, P. Avgeriou, and Y. Li, “Architectural design decisions that incur technical debt—an industrial case study”, *Information and Software Technology*, vol. 139, p. 106 669, 2021.

[26] A. Zalewski, K. Borowa, and A. Ratkowski, “On cognitive biases in architecture decision making”, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10475 LNCS, 2017, pp. 123–137, ISBN: 9783319658308. DOI: 10.1007/978-3-319-65831-59.

[27] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing technical debt in software engineering (dagstuhl seminar 16162)”, in *Dagstuhl reports*, Schloss Dagstuhl - Leibniz - Zentrum fuer Informatik, vol. 6, 2016.

[28] T. Besker, A. Martini, and J. Bosch, “Managing architectural technical debt: A unified model and systematic literature review”, *Journal of Systems and Software*, vol. 135, pp. 1–16, 2018, ISSN: 01641212. DOI: 10.1016/j.jss.2017.09.025. [Online]. Available: https://doi.org/10.1016/j.jss.2017.09.025.

[29] M. Shepperd, C. Mair, and M. Jørgensen, “An Experimental Evaluation of a De-biasing Intervention for Professional Software Developers”, in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018. DOI: 10.1145/3167132.3167293. arXiv: 1804.03919. [Online]. Available: http://arxiv.org/abs/1804.03919%7B%5C%%7D0Ahttp://dx.doi.org/10.1145/3167132.3167293.

[30] K. M. Borowa, “On cognitive biases in software engineering”, Instytut Automatyki i Informatyki Stosowanej, 2019.

[31] W. James, *Principles of Psychology (1980)*. Henry Holt and Company, 1931.

[32] M. I. Posner, C. R. Snyder, and R. Solso, “Attention and cognitive control”, *Cognitive psychology: Key readings*, vol. 205, pp. 55–85, 2004.

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

[34] D. Kahneman and A. Tversky, “Subjective probability: A judgment of representativeness”, *Cognitive psychology*, vol. 3, no. 3, pp. 430–454, 1972.

[35] T. Pachur, R. Hertwig, and F. Steinmann, “How do people judge risks: Availability heuristic, affect heuristic, or both?”, *Journal of Experimental Psychology: Applied*, vol. 18, no. 3, p. 314, 2012.

[36] A. Furnham and H. C. Boo, “A literature review of the anchoring effect”, *The journal of socio-economics*, vol. 40, no. 1, pp. 35–42, 2011.

[37] W. Wattanacharoensil and D. La-ornual, “A systematic review of cognitive biases in tourist decisions”, *Tourism Management*, vol. 75, pp. 353–369, 2019.

[38] G. Saposnik, D. Redelmeier, C. C. Ruff, and P. N. Tobler, “Cognitive biases associated with medical decisions: A systematic review”, *BMC medical informatics and decision making*, vol. 16, no. 1, pp. 1–14, 2016.

[39] M. E. Oswald and S. Grosjean, “Confirmation bias”, in *Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory*, R. Pohl, Ed., Psychology Press, 2004, pp. 97–114.

[40] L. M. Leventhal, B. M. Teasley, D. S. Rohlman, and K. Instone, “Positive test bias in software testing among professionals: A review”, in *Human-Computer Interaction: Third International Conference, EWHCI’93 Moscow, Russia, August 3–7, 1993 Selected Papers 3*, Springer, 1993, pp. 210–218.

[41] A. Bracha and D. J. Brown, “Affective decision making: A theory of optimism bias”, *Games and Economic Behavior*, vol. 75, no. 1, pp. 67–80, 2012.

[42] R. Buehler, D. Griffin, and M. Ross, “Exploring the" planning fallacy": Why people underestimate their task completion times.”, *Journal of personality and social psychology*, vol. 67, no. 3, p. 366, 1994.

[43] M. Ross and F. Sicoly, “Egocentric biases in availability and attribution.”, *Journal of personality and social psychology*, vol. 37, no. 3, p. 322, 1979.

[44] S. Freud, *Three essays on the theory of sexuality: The 1905 edition*. Verso Books, 2017.

[45] P. Ralph, “Toward a theory of debiasing software development”, *Lecture Notes in Business Information Processing*, vol. 93 LNBIP, pp. 92–105, 2011, ISSN: 18651348. DOI: 10.1007/978-3-642-25676-9_8.

[46] D. J. Malenka, J. A. Baron, S. Johansen, J. W. Wahrenberger, and J. M. Ross, “The framing effect of relative and absolute risk”, *Journal of general internal medicine*, vol. 8, pp. 543–548, 1993.

[47] J. L. Nicolau, J. P. Mellinas, and E. Martín-Fuentes, “The halo effect: A longitudinal approach”, *Annals of Tourism Research*, vol. 83, p. 102 938, 2020.

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

[49] M. G. Pitts and G. J. Browne, “Improving requirements elicitation: An empirical investigation of procedural prompts”, *Information systems journal*, vol. 17, no. 1, pp. 89–110, 2007.

[50] S. Chakraborty, S. Sarker, and S. Sarker, “An exploration into the process of requirements elicitation: A grounded approach”, *Journal of the association for information systems*, vol. 11, no. 4, p. 1, 2010.

[51] A. Zalewski, K. Borowa, and D. Kowalski, “On cognitive biases in requirements elicitation”, in *Integrating Research and Practice in Software Engineering*, Springer, 2020, pp. 111–123.

[52] W. Stacy and J. MacMillan, “Cognitive bias in software engineering”, *Communications of the ACM*, vol. 38, no. 6, pp. 57–63, 1995.

[53] J. Parsons and C. Saunders, “Cognitive heuristics in software engineering applying and extending anchoring and adjustment to artifact reuse”, *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 873–888, 2004.

[54] G. Allen and J. Parsons, “Is query reuse potentially harmful? anchoring and adjustment in adapting existing database queries”, *Information Systems Research*, vol. 21, no. 1, pp. 56–77, 2010.

[55] G. Calikli and A. Bener, “Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance”, in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010, pp. 1–11.

[56] G. Çalıklı and A. B. Bener, “Influence of confirmation biases of developers on software quality: An empirical study”, *Software Quality Journal*, vol. 21, pp. 377–416, 2013.

[57] G. Calikli, A. Bener, and B. Arslan, “An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers”, in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, 2010, pp. 187–190.

[58] G. Calikli, B. Aslan, and A. Bener, “Confirmation bias in software development and testing: An analysis of the effects of company size, experience and reasoning skills”, 2010.

[59] M. Jørgensen and D. I. Sjøberg, “Software process improvement and human judgement heuristics”, *Scandinavian Journal of Information Systems*, vol. 13, no. 1, p. 2, 2001.

[60] K. Moløkken and M. Jørgensen, “Software effort estimation: Unstructured group discussion as a method to reduce individual biases.”, in *PPIG*, 2003, p. 4.

[61] K. Moløkken-Østvold and M. Jørgensen, “Group processes in software effort estimation”, *Empirical Software Engineering*, vol. 9, no. 4, pp. 315–334, 2004.

[62] K. Molokken-Ostvold and N. C. Haugen, “Combining estimates with planning poker–an empirical study”, in *2007 Australian Software Engineering Conference (ASWEC’07)*, IEEE, 2007, pp. 349–358.

[63] K. Moløkken and M. Jørgensen, “Expert estimation of the effort of web-development projects: Why are software professionals in technical roles more optimistic than those in nontechnical roles”, *Journal of Empirical Software Engineering*, 2004.

[64] M. Jørgensen, K. H. Teigen, and K. Moløkken, “Better sure than safe? over-confidence in judgement based software development effort prediction intervals”, *Journal of systems and software*, vol. 70, no. 1-2, pp. 79–93, 2004.

[65] M. Jorgensen and S. Grimstad, “Over-optimism in software development projects:" the winner’s curse"”, in *15th International Conference on Electronics, Communications and Computers (CONIELECOMP’05)*, IEEE, 2005, pp. 280–285.

[66] M. Jørgensen and B. Faugli, “Prediction of overoptimistic predictions”, in *10th International Conference on Evaluation and Assessment in Software Engineering (EASE) 10*, 2006, pp. 1–10.

[67] M. Jørgensen, “Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates”, *Information and Software Technology*, vol. 52, no. 5, pp. 506–516, 2010.

[68] O. Shmueli, N. Pliskin, and L. Fink, “Can the outside-view approach improve planning decisions in software development projects?”, *Information Systems Journal*, vol. 26, no. 4, pp. 395–418, 2016.

[69] K. M. Lui and K. C. Chan, “A cognitive model for solo programming and pair programming”, in *Proceedings of the Third IEEE International Conference on Cognitive Informatics, 2004.*, IEEE, 2004, pp. 94–102.

[70] F. Ramin, “The role of egocentric bias in undergraduate agile software development teams”, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, 2020, pp. 122–124.

[71] P. Tobias and D. S. Spiegel, *Is design the preeminent protagonist in user experience?*, 2009.

[72] C. Gacek, A. Abd-Allah, B. Clark, and B. Boehm, “On the definition of software system architecture”, in *Proceedings of the First International Workshop on Architectures for Software Systems*, Seattle, Wa, 1995, pp. 85–94.

[73] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[74] A. Tang and R. Kazman, “Decision-making principles for better software design decisions”, *IEEE Software*, vol. 38, no. 6, pp. 98–102, 2021.

[75] A. Tang, M. Razavian, B. Paech, and T. M. Hesse,

[76] A. Tang and H. van Vliet, “Design strategy and software design effectiveness”, *IEEE software*, vol. 29, no. 1, pp. 51–55, 2011.

[77] H. van Vliet and A. Tang, “Decision making in software architecture”, *Journal of Systems and Software*, vol. 117, pp. 638–644, 2016, ISSN: 01641212. DOI: `10.1016/j.jss.2016.01.017`.

[78] B. Fischhoff, “Debiasing. judgment under uncertainty: Heuristics and biases”, *Judgment under uncertainty: Heuristics and biases*, pp. 422–444, 1982.

[79] M. Razavian, A. Tang, R. Capilla, and P. Lago, “In two minds: How reflections influence software design thinking”, *Journal of Software: Evolution and Process*, vol. 28, no. 6, pp. 394–426, 2016.

[80] A. Tang, F. Bex, C. Schriek, and J. M. E. van der Werf, “Improving software design reasoning–a reminder card approach”, *Journal of Systems and Software*, vol. 144, pp. 22–40, 2018.

[81] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, “Managing architectural decision models with dependency relations, integrity constraints, and production rules”, *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249–1267, 2009.

[82] A. Tang, M. A. Babar, I. Gorton, and J. Han, “A survey of architecture design rationale”, *Journal of Systems and Software*, vol. 79, no. 12, pp. 1792–1804, 2006, ISSN: 01641212.

[83] M. Razavian, B. Paech, and A. Tang, “Empirical research for software architecture decision making: An analysis”, *Journal of Systems and Software*, vol. 149, pp. 360–381, 2019, ISSN: 01641212.

[84] R. Weinreich, I. Groher, and C. Miesbauer, “An expert survey on kinds, influence factors and documentation of design decisions in practice”, *Future Generation Computer Systems*, vol. 47, pp. 145–160, 2015, ISSN: 0167739X.

[85] C. Miesbauer and R. Weinreich, “Classification of design decisions–an expert survey in practice”, in *Software Architecture: 7th European Conference, ECSA 2013, Montpellier, France, July 1-5, 2013. Proceedings 7*, Springer, 2013, pp. 130–145.

[86] T. Bi, P. Liang, and A. Tang, “Architecture Patterns, Quality Attributes, and Design Contexts: How Developers Design with Them”, *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2018-Decem, no. 61472286, pp. 49–58, 2018, ISSN: 15301362.

[87] M. Soliman, M. Riebisch, and U. Zdun, “Enriching Architecture Knowledge with Technology Design Decisions”, *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, pp. 135–144, 2015.

[88] M. Bhat, K. Shumaiev, U. Hohenstein, A. Biesdorf, and F. Matthes, “The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study”, in *2020 IEEE International Conference on Software Architecture (ICSA)*, IEEE, 2020, pp. 69–80.

[89] T. Bi, P. Liang, A. Tang, and X. Xia, “Mining Architecture Tactics and Quality Attributes Knowledge in Stack Overflow”, *Journal of Systems and Software*, no. May, 2021.

[90] ISO/IEC 25010, *ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models*, 2011.

[91] M. Bhat, C. Tinnes, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, “Adex: A tool for automatic curation of design decision knowledge for architectural decision recommendations”, in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, IEEE, 2019, pp. 158–161.

[92] M. X. Liu, J. Hsieh, N. Hahn, *et al.*, “Unakite: Scaffolding developers’ decision-making using the web”, in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019, pp. 67–80.

[93] M. Razavian, A. Tang, R. Capilla, and P. Lago, “Reflective approach for software design decision making”, *Proceedings - 1st Workshop on Qualitative Reasoning about Software Architectures, QRASA 2016*, pp. 19–26, 2016.

[94] A. Tang and R. Kazman, "Decision-Making Principles for Better Software Design Decisions", *IEEE Software*, vol. 38, no. 6, pp. 98–102, 2021, ISSN: 19374194.

[95] J. E. Burge, "Design rationale: Researching under uncertainty", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 22, no. 4, pp. 311–324, 2008, ISSN: 08900604.

[96] M. J. De Dieu, P. Liang, and M. Shahin, "How Do Developers Search for Architectural Information? An Industrial Survey", *Proceedings - IEEE 19th International Conference on Software Architecture, ICSA 2022*, no. December 2021, pp. 58–68, 2022. arXiv: `2112.10920`.

[97] D. Tofan, M. Galster, and P. Avgeriou, "Difficulty of architectural decisions – a survey with professional architects", in *Software Architecture*, K. Drira, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 192–199, ISBN: 978-3-642-39031-9.

[98] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. 2012, ISBN: 9781118104354. [Online]. Available: `www.wiley.com.`.

[99] K. Borowa, R. Lewanczyk, K. Stpiczyńska, P. Stradomski, and A. Zalewski, *What rationales drive architectural decisions? An empirical inquiry - Additional material*, version 1, Zenodo, May 2023. DOI: `10.5281/zenodo.7946764`. [Online]. Available: `https://doi.org/10.5281/zenodo.7946764`.

[100] J. Saldaña, "The coding manual for qualitative researchers", *The coding manual for qualitative researchers*, pp. 1–440, 2021.

[101] D. R. Garrison, M. Cleveland-Innes, M. Koole, and J. Kappelman, "Revisiting methodological issues in transcript analysis: Negotiated coding and reliability", *Internet and Higher Education*, vol. 9, no. 1, pp. 1–8, 2006, ISSN: 10967516.

[102] K. Beck, M. Beedle, A. van Bennekum, *et al.* "Principles behind the Agile Manifesto". (2001), [Online]. Available: `https://agilemanifesto.org/principles.html` (visited on 07/05/2021).

[103] P. Kruchten, R. Nord, and I. Ozkaya, *Managing Technical Debt*. Addison-Wesley Professional, 2019.

[104] A. Zalewski, K. Borowa, and A. Ratkowski, "On cognitive biases in architecture decision making", in *Software Architecture*, A. Lopes and R. de Lemos, Eds., Cham: Springer International Publishing, 2017, pp. 123–137, ISBN: 978-3-319-65831-5.

[105] K. Daniel, *Thinking, fast and slow*. 2017.

[106] P. Naur and B. Randell, "Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7th-11th october 1968", 1969.

[107] J. N. Buxton and B. Randell, *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee*. NATO Science Committee; available from Scientific Affairs Division, NATO, 1970.

[108] A. Tversky and D. Kahneman, "Rational choice and the framing of decisions", *Decision making: Descriptive, normative, and prescriptive interactions*, pp. 167–192, 1988.

[109] H. Leibenstein, "Bandwagon, snob, and veblen effects in the theory of consumers' demand", *The quarterly journal of economics*, vol. 64, no. 2, pp. 183–207, 1950.

[110] S. A. Birch and P. Bloom, "The curse of knowledge in reasoning about false beliefs", *Psychological Science*, vol. 18, no. 5, pp. 382–386, 2007.

[111] D. Kahneman and J. Renshon, "Hawkish biases", *American Foreign Policy and the Politics of Fear: Threat Inflation Since*, vol. 9, no. 11, pp. 79–96, 2009.

[112] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers", *Information and Software Technology*, vol. 49, no. 6, pp. 637–653, 2007.

[113] A. Tang and H. van Vliet, "Software designers satisfice", in *Software Architecture: 9th European Conference, ECSA 2015, Dubrovnik/Cavtat, Croatia, September 7-11, 2015. Proceedings 9*, Springer, 2015, pp. 105–120.

[114] J. Kruger and D. Dunning, "Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments.", *Journal of personality and social psychology*, vol. 77, no. 6, p. 1121, 1999.

[115] W. H. Brown, R. C. Malveau, H. W. S. McCormick, and T. J. Mowbray, *AntiPatterns: refactoring software, architectures, and projects in crisis*. John Wiley & Sons, Inc., 1998.

[116] R. S. Nickerson, "Confirmation bias: A ubiquitous phenomenon in many guises", *Review of general psychology*, vol. 2, no. 2, pp. 175–220, 1998.

[117] M. I. Norton, D. Mochon, and D. Ariely, "The ikea effect: When labor leads to love", *Journal of consumer psychology*, vol. 22, no. 3, pp. 453–460, 2012.

[118] C. N. Parkinson, *Parkinson's Law, or the Pursuit of Progress*. 1958.

[119] E. M. Rogers, *Diffusion of innovations*. Simon and Schuster, 2010.

[120] M. H. Bazerman and M. A. Neale, *Negotiating rationally*. Simon and Schuster, 1993.

[121] T. Sharot, A. M. Riccardi, C. M. Raio, and E. A. Phelps, "Neural mechanisms mediating optimism bias", *Nature*, vol. 450, no. 7166, pp. 102–105, 2007.

[122] N. Rios, M. G. de Mendonça Neto, and R. O. Spínola, "A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners", *Information and Software Technology*, vol. 102, pp. 117–145, 2018.

[123] W. Cunningham, "The WyCash portfolio management system", in *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, vol. Part F1296, 1992, pp. 29–30, ISBN: 0897916107. DOI: 10.1145/157709.157715.

[124] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt", *Journal of Systems and Software*, vol. 86, no. 6, pp. 1498–1516, 2013.

[125] T. Amanatidis, N. Mittas, A. Chatzigeorgiou, A. Ampatzoglou, and L. Angelis, "The developer's dilemma: Factors affecting the decision to repay code debt", in *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, vol. 5, 2018, pp. 62–66, ISBN: 9781450357135. DOI: 10.1145/3194164.3194174. [Online]. Available: https://doi.org/10.1145/3194164.3194174.

[126] A. Martini and J. Bosch, "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles", in *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, 2015, pp. 1–10, ISBN: 9781479919222. DOI: 10.1109/WICSA.2015.31. [Online]. Available: https://www.researchgate.net/publication/273769557.

[127] R. Verdecchia, P. Kruchten, and P. Lago, "Architectural Technical Debt : A Grounded Theory", *European Conference on Software Architecture (ECSA)*, 2020.

[128] R. Brenner, "Balancing resources and load: Eleven nontechnical phenomena that contribute to formation or persistence of technical debt", in *Proceedings - 2019 IEEE/ACM International Conference on Technical Debt, TechDebt 2019*, 2019, pp. 38–47, ISBN: 9781728133713. DOI: 10.1109/TechDebt.2019.00013.

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

[130] J. Bosch, "Software architecture: The next step", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioin-*

formatics), vol. 3047, pp. 194–199, 2004, ISSN: 03029743. DOI: 10.1007/978-3-540-24769-214.

[131] R. Alfayez, W. Alwehaibi, R. Winn, E. Venson, and B. Boehm, "A systematic literature review of technical debt prioritization", in *Proceedings of the 3rd International Conference on Technical Debt*, vol. 10, ACM, 2020, pp. 1–10, ISBN: 9781450379601. DOI: 10.1145/3387906.3388630. [Online]. Available: https://doi.org/10.1145/3387906.3388630.

[132] C. Becker, R. Chitchyan, S. Betz, and C. McCord, "Trade-off decisions across time in technical debt management: A systematic literature review", in *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, ACM, 2018, pp. 85–94, ISBN: 9781450357135. DOI: 10.1145/3194164.3194171. [Online]. Available: https://doi.org/10.1145/3194164.3194171.

[133] N. Rios, M. Mendonça, and R. Spínola, "A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners", *Information and Software Technology*, vol. 102, Jun. 2018. DOI: 10.1016/j.infsof.2018.05.010.

[134] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: Understanding causes and a qualitative model", in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, 2014, pp. 85–92.

[135] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? Software practitioners and technical debt", in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, 2015, pp. 50–60, ISBN: 9781450336758. DOI: 10.1145/2786805.2786848. [Online]. Available: http://github.com/neilernst/td-survey.

[136] A. Tversky and D. Kahneman, "The framing of decisions and the psychology of choice", *Science*, 1981, ISSN: 00368075. DOI: 10.1126/science.7455683.

[137] G. B. Chapman and B. H. Bornstein, "The more you ask for, the more you get: Anchoring in personal injury verdicts", *Applied Cognitive Psychology*, 1996, ISSN: 08884080. DOI: 10.1002/(SICI)1099-0720(199612)10:6<519::AID-ACP417>3.0.CO;2-5.

[138] J. Kennedy, "Debiasing in the Audit Curse of Knowledge Judgment", *The Accounting Review*, 1995, ISSN: 00014826.

[139] M. I. Norton, D. Mochon, and D. Ariely, "The IKEA effect: When labor leads to love", *Journal of Consumer Psychology*, 2012, ISSN: 10577408. DOI: `10.1016/j.jcps.2011.08.002`.

[140] C. Northcote Parkinson, "Parkinson's law: Or the pursuit of progress", 1961.

[141] E. M. Rogers, A. Singhal, and M. M. Quinlan, "Diffusion of innovations", in *An Integrated Approach to Communication Theory and Research, Third Edition*, 2019, ISBN: 9781351358712. DOI: `10.4324/9780203710753-35`.

[142] M. V. Pezzo, J. A. Litman, and S. P. Pezzo, "On the distinction between yuppies and hippies: Individual differences in prediction biases for planning future tasks", *Personality and Individual Differences*, 2006, ISSN: 01918869. DOI: `10.1016/j.paid.2006.03.029`.

[143] H. Leibenstein, "Bandwagon, snob, and veblen effects in the theory of consumers' demand", *Quarterly Journal of Economics*, 1950, ISSN: 15314650. DOI: `10.2307/1882692`.

[144] B. M. Staw, "The escalation of commitment: An update and appraisal", in *Organizational Decision Making*, 2010. DOI: `10.1017/cbo9780511584169.011`.

[145] A. H. Maslow, *The psychology of science; a reconnaissance.* 1966, ISBN: National Library: 0354146 LCCN: 66-11479.

[146] O. P. O'Sulliivan, "The Neural Basis of Always Looking on the Bright Side", *Dialogues in Philosophy, Mental and Neuro*, 2015.

[147] T. Besker, A. Martini, and J. Bosch, "Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work", in *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, vol. 10, 2018, pp. 105–114, ISBN: 9781450357135. DOI: `10.1145/3194164.3194178`. [Online]. Available: `https://doi.org/10.1145/3194164.3194178`.

[148] R. Verdecchia, "Architectural Technical Debt Identification: Moving Forward", *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, pp. 43–44, 2018. DOI: `10.1109/ICSA-C.2018.00018`.

[149] M. M. Lehman, “Programs, Life Cycles, and Laws of Software Evolution”, *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, 1980, ISSN: 15582256. DOI: 10.1109/PROC.1980.11805.

[150] T. Stablein, D. Berndt, and M. Mullarkey, “Technical debt-related information asymmetry between finance and IT”, in *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 134–137, ISBN: 9781450357135. DOI: 10.1145/3194164.3194180. [Online]. Available: https://doi.org/10.1145/3194164.3194180.

[151] T. Besker, A. Martini, R. Edirisooriya Lokuge, K. Blincoe, and J. Bosch, “Embracing technical debt, from a startup company perspective”, *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018*, pp. 415–425, 2018. DOI: 10.1109/ICSME.2018.00051.

[152] T. Besker, A. Martini, and J. Bosch, “Carrot and stick approaches when managing technical debt”, in *Proceedings of the 3rd International Conference on Technical Debt*, 2020, pp. 21–30, ISBN: 9781450379601. DOI: 10.1145/3387906.3388619. [Online]. Available: https://doi.org/10.1145/3387906.3388619.

[153] M. E. Fonteyn, B. Kuipers, and S. J. Grobe, “A description of think aloud method and protocol analysis”, *Qualitative health research*, vol. 3, no. 4, pp. 430–441, 1993.

[154] G. Çalikli and A. B. Bener, “Influence of confirmation biases of developers on software quality: An empirical study”, *Software Quality Journal*, vol. 21, no. 2, pp. 377–416, 2013, ISSN: 09639314. DOI: 10.1007/s11219-012-9180-0.

[155] A. Jansen and J. Bosch, “Software architecture as a set of architectural design decisions”, in *Proceedings - 5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005*, vol. 2005, 2005, pp. 109–120, ISBN: 0769525482.

[156] K. Borowa, A. Zalewski, and S. Kijas, “The Influence of Cognitive Biases on Architectural Technical Debt”, in *International Conference on Software Architecture (ICSA)*, 2021.

[157] A. Tang, F. Bex, C. Schriek, and J. M. E. van der Werf, “Improving software design reasoning–A reminder card approach”, *Journal of Systems and Software*, vol. 144, no. April 2017, pp. 22–40, 2018, ISSN: 01641212. DOI: 10.1016/j.jss.2018.05.019. [Online]. Available: https://doi.org/10.1016/j.jss.2018.05.019.

[158] W. Stacy and J. Macmillan, “Cognitive Bias in Software Engineering”, *Communications of the ACM*, vol. 38, no. 6, pp. 57–63, 1995, ISSN: 15577317. DOI: 10.1145/203241.203256.

[159] K. Borowa, R. Dwornik, and A. Zalewski, “Is knowledge the key? an experiment on debiasing architectural decision-making-a pilot study”, in *International Conference on Product-Focused Software Process Improvement*, Springer, 2021, pp. 207–214.

[160] A. Tversky and Kahneman Daniel, “Judgment under uncertainty: Heuristics and biases”, *Science*, 1974, ISSN: 15206882.

[161] K. Borowa, M. Jarek, G. Mystkowska, W. Paszko, and A. Zalewski, *Additional Material for Debiasing architectural decision-making: a workshop-based training approach*, Zenodo, Jun. 2022. DOI: 10.5281/zenodo.6751990. [Online]. Available: https://doi.org/10.5281/zenodo.6751990.

[162] K. Borowa, A. Zalewski, and S. Kijas, “The influence of cognitive biases on architectural technical debt”, in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, IEEE, 2021, pp. 115–125.

[163] S. Brown, *The c4 model for software architecture*, Jun. 2018. [Online]. Available: https://www.infoq.com/articles/C4-architecture-model/.

[164] J. Saldaña, “The coding manual for qualitative researchers”, *The coding manual for qualitative researchers*, pp. 1–440, 2021.

[165] G. R. Norman, S. D. Monteiro, J. Sherbino, J. S. Ilgen, H. G. Schmidt, and S. Mamede, “The causes of errors in clinical reasoning: Cognitive biases, knowledge deficits, and dual process thinking”, *Academic Medicine*, vol. 92, no. 1, pp. 23–30, 2017.

[166] K. Borowa, M. Jarek, G. Mystkowska, W. Paszko, and A. Zalewski, “Debiasing architectural decision-making: a workshop-based training approach”, in *European Conference on Software Architecture (ECSA)*, 2022, pp. 1–8. arXiv: 2206.14701. [Online]. Available: http://arxiv.org/abs/2206.14701.

[167] M. Galster and D. Weyns, “Empirical research in software architecture—perceptions of the community”, *Journal of Systems and Software*, vol. 202, p. 111 684, 2023.

[168] S. Baltes and P. Ralph, “Sampling in software engineering research: A critical review and guidelines”, *Empirical Software Engineering*, vol. 27, no. 4, p. 94, 2022.

[169] U. Van Heesch, P. Avgeriou, and R. Hilliard, “A documentation framework for architecture decisions”, *Journal of Systems and Software*, vol. 85, no. 4, pp. 795–820, 2012.

[170] A. Manjunath, M. Bhat, K. Shumaiev, A. Biesdorf, and F. Matthes, "Decision Making and Cognitive Biases in Designing Software Architectures", *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, pp. 52–55, 2018. DOI: `10.1109/ICSA-C.2018.00022`.

[171] K. Borowa, A. Zalewski, and S. Kijas, "The Influence of Cognitive Biases on Architectural Technical Debt", in *International Conference on Software Architecture (ICSA)*, 2021.

[172] E. Løhre and M. Jørgensen, "Numerical anchors and their strong effects on software development effort estimates", *Journal of Systems and Software*, vol. 116, pp. 49–56, 2016, ISSN: 01641212. DOI: `10.1016/j.jss.2015.03.015`.

[173] T. Sharot, "The optimism bias", *Current biology*, vol. 21, no. 23, R941–R945, 2011.

[174] E. Shalev, M. Keil, J. S. Lee, and Y. Ganzach, "Optimism Bias in Managing It Project Risks: a Construal Level", *European Conference on Information Systems*, 2014.

[175] C. W. Turner, J. R. Lewis, and J. Nielsen, "Determining usability test sample size", *International encyclopedia of ergonomics and human factors*, vol. 3, no. 2, pp. 3084–3088, 2006.

[176] F. Wilcoxon, "Individual comparisons by ranking methods", in *Breakthroughs in statistics: Methodology and distribution*, Springer, 1992, pp. 196–202.

[177] K. A. Ericsson and H. A. Simon, *Protocol analysis: Verbal reports as data, Rev. ed.* Cambridge, MA, US: The MIT Press, 1993, pp. liii, 443–liii, 443, ISBN: 0-262-05047-1 (Hardcover); 0-262-55023-7 (Paperback).

[178] Anonymous, *Additional Material for Debiasing Architectural Decision-Making: Teaching Software Practitioners*, Zenodo, Apr. 2024. DOI: `10.5281/zenodo.11047872`.

[179] M. Soliman, M. Wiese, Y. Li, M. Riebisch, and P. Avgeriou, "Exploring web search engines to find architectural knowledge", in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, IEEE, 2021, pp. 162–172.

[180] H. Cervantes and R. Kazman, *Designing software architectures: a practical approach*. Addison-Wesley Professional, 2016.

[181] H. Muccini *et al.*, "Group decision-making in software architecture: A study on industrial practices", *Information and software technology*, vol. 101, pp. 51–63, 2018.

[182] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction (International Series in Software*

Engineering). Springer Science & Business Media, 2000, ISBN: 978-1-4615-4625-2. [Online]. Available: `https://www.springer.com/gp/book/9781461370918%7B%5C#%7DaboutBook`.

List of Figures

1.1 Thesis overview 21
2.1 Dual-process judgment 25
3.1 Study phases 39
3.2 Questionnaire participants 40
3.3 Questionnaire participants companies 40
4.1 How biases influence architecture decision making 60
6.1 Biased arguments 102
6.2 Argument count 102
6.3 Biases in statements 103
8.1 Study steps 125
8.2 Average code sums for each measured value 131

List of Tables

3.1 Interview participants 41
3.2 Codes 43
3.3 Questionnaire results. ISO/IEC 25010 quality attributes are marked by a **bold** font. 44
4.1 Biases relevant to architecture decision making indicated by the workshop participants 59
4.2 Influence of cognitive biases on architecture decision making 64
5.1 Participant data 79
5.2 Qualitative analysis codes 80
5.3 Technical debt occurrences mentioned by participants 81
5.4 Cognitive biases present in the participants' accounts 83
5.5 Cognitive biases influencing ATD items 84
6.1 Coding Scheme 101
6.2 Participant data 101
8.1 Participants (W-workshop group / C-Control group) 123
8.2 Coding scheme: adapted from [166] 130
8.3 p-values for each measurement 132

8.4 (Non-)Biased Arguments and Counterarguments(B – Biased, NB – Non-biased) 133
8.5 Occurrences of Cognitive Bias (ANCH – Anchoring, OPT – Optimism bias, CONF – Confirmation bias) 134
8.6 Use of Debiasing techniques (DRAW – Stating drawback, MULTI – Listing solution alternatives , RISK – Stating risks) 135
8.7 Decisions discussed 135