COGNITIVE BIAS IN SOFTWARE DEVELOPMENT: A THEMATIC LITERATURE REVIEW OF DEBIASING STRATEGIES

Lappeenranta–Lahti University of Technology LUT

Bachelor's thesis, Software Engineering

2026

Author: Pranita Dangi

Supervisor: M.Sc. (Tech.) Kseniia Suomalainen

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Software Engineering

Pranita Dangi

Cognitive Bias in Software Development: A Thematic Literature Review of Debiasing Strategies

Bachelor's thesis

2026

33 pages and 5 figures

Supervisor: M.Sc. (Tech.) Kseniia Suomalainen

Keywords: cognitive bias, software development, testing, decision-making, thematic analysis.

Software development involves many activities that require human decision-making in the process, such as design, estimation, testing, and code review. When programmers and testers work on these activities, they tend to use mental shortcuts referred to as cognitive biases.

This bachelor's thesis explores the problem of cognitive bias in software development and simple strategies for minimizing the negative impact of cognitive bias. The study aims to gain insight into how cognitive bias affects everyday software development activities and how supporting approaches based on awareness can aid in improved decision-making.

The research was carried out by a literature review method with a thematic analysis technique. Academic literature and industry sources were gathered, coded, and organized into themes concerning the cognitive and social biases and de-biasing methods. Furthermore, a brief exercise to test for bias awareness was conducted: a basic Python program was tested prior to and after learning about confirmation bias.

Results suggest that there is a cognitive bias impact in software development activities, especially software testing and decision-making. Themes identified were awareness and training methods; team and process practices. The practical exercise also found that greater awareness fostered wider testing as it brought attention to edge cases and other inputs that weren't considered obvious. This study has its limitations, as it is dependent on literature and a tiny individual consciousness exercise. So, the conclusions do not apply to all software development teams. Larger development teams and the long-term effects of awareness-based debiasing techniques in actual software projects could be considered for future studies.

Table of contents

Abstract

Declarations

1 Introduction 6
1.1 Research Aim and Objectives 7
1.2 Structure of the Thesis 7
2 Cognitive Bias in Software Development: Literature Review 9
2.1 Learning Cognitive Bias 9
2.1.1 Defining the nature of Bias 9
2.1.2 General Biases of Software Projects 10
2.2 Impact of Biases on Software Quality 11
2.3 Expectancies of Debiasing at this moment 12
2.4 Debiasing Tools, Techniques, and Frames in Software Development 13
3 Research Methodology 14
3.1 Research Design 14
3.2 Choice of Thematic Analysis 14
3.3 Data Collection Process 15
3.3.1 Selection of Sources 15
3.4 Awareness Exercise Procedure 16
3.5 Considerations of Reliability, Validity, and Research Quality 17
4 Findings 18
4.1 Theme 1: Awareness and Training Methods 18
4.2 Small Bias-Awareness Testing Exercise 18
4.2.1 First Testing Phase 19
4.2.2 Second Testing Phase 20
4.2.3 Visual comparison of the exercise 22
4.3 Theme 2: Team and Process Practices 23
4.4 Summary of Findings 24
4.4.1 Summary of Theme 1 24
4.4.2 Summary of Theme 2 25
4.4.3 Relationship Between Themes 25

5 Discussion 27
5.1 Interpretation of Findings 27
5.2 Comparison to the preceding Reports 28
5.3 Practical Implications on the Software Development of Teams 28
5.4 Limitations of the Study 30
5.5 Future Research 30
6 Conclusion 31
References 32

DECLARATIONS

Turnitin

The originality of this thesis has been reviewed with the Turnitin similarity checking service.

AI usage

The author of this thesis, Pranita Dangi, used the following AI tools during the preparation of the thesis:

1. AI tool name: ChatGPT (OpenAI)
    a. Purpose of use: Language refinement, phrasing, structural clarity, and reference formatting.
    b. Explanation of the use of the tool: ChatGPT was used to help rephrase sentences to improve readability and to help format the references according to the Harvard style. It was not created to develop research content or analysis or to draw conclusions.
2. AI tool name: QuillBot
    a. Purpose of use: Grammar checking, spelling correction, and paraphrasing for clarity.
    b. Explanation of the use of the tool: QuillBot was used to edit grammar, correct misspelled words, and provide clearer sentence structures. Only the free version was used. All suggestions were considered and accepted by the author.

Responsibility

The author, Pranita Dangi, declares that this thesis is my own work and that it has not been copied from any other source. All the research, analysis, findings, and conclusions are the responsibility of the author. All AI-assisted texts have been thoroughly reviewed and verified by the author.

1 Introduction

Software development is regarded as a technical process, but it highly relies on human thought and judgment. There are numerous decisions that developers and testers must make daily. These involve choosing design features, determining the duration of the work, testing software, and reviewing code written by others. Many such choices are made in the face of time and with scanty information. When human beings make judgments, they do not always analyze everything clearly. On the contrary, they tend to apply mental shortcuts. These are cognitive biases that are shortcuts. Cognitive bias refers to the fact that individuals are predisposed to thinking in some way, and this may at times result in wrong assumptions. In computer programming, it can lead to developers or testers believing that their original idea is too good, concentrating on what was expected, or too much dependence on tools and power.

Research indicates that cognitive bias is manifested in a wide variety of fields within the scope of software engineering. Software estimation, debugging, testing, and code review have been found to have bias. As an example, testers can cease to test when they observe anticipated outputs, or developers can use initial estimates when new data emerges. Similar issues in the industry are also covered by the blogs of software engineers and surveys of developers. Since software systems are common in society, inaccuracies that are occasioned by biased decisions may have severe consequences. This makes cognitive bias a significant subject for research on software development.

Why cognitive bias is important: Software is ubiquitous in today's world. Whether it's in the bank app, at the hospital, or even in the pocket of anyone using a mobile phone, software has an impact on nearly everything people do. Bugs and bad design decisions can result from poor and biased decision-making and can cause anything from a few minor glitches to dangerous safety issues. Therefore, it is important to understand more than just the technical aspects of software development and to understand the human aspects of software development. Developers and testers can understand how to mitigate the impact of those cognitive biases and develop more reliable software and make better decisions. In this thesis, it is hoped that this new understanding of the existence of bias in software development and some simple suggestions to correct it will help towards that objective. The first step to reducing the impact of cognitive biases is understanding them. This thesis examines the biases and methods of reducing biases.

1.1 Research Aim and Objectives

This thesis aims to examine cognitive bias in software development and its effects on software quality and team decision-making. Testing, estimation, and code review are software development activities that usually require judgment. Cognitive and social biases can influence these judgments, causing defects to be overlooked, errors in estimating, and unjust judgments. The primary objective of the thesis is to find out the most typical cognitive biases in software development and to study how these negative impacts can be minimized in the simplest possible way. The work is premised on a literature survey and a mini-awareness-based event that was influenced by the mid-term thesis presentation. The three research questions included are:

- What are the cognitive and social biases in the software development activities?
- What impact do these biases have on the quality of the software and decision-making?
- Will awareness-based interventions prevent bias in software development?

This thesis particularly aims to investigate specific cognitive biases that are found in common software development tasks such as testing, estimation, and code review. Research in cognitive bias has been conducted in software engineering, but most focus on specific cognitive biases in isolation. This study is an attempt to have a more comprehensive perspective and to examine several forms of bias and their influence on one another and on team dynamics. The purpose is not only to explain the issues, but to come up with a solution that is useful to the software teams. The thesis is designed to demonstrate the potential impact on the way people test and make decisions when they are only aware of a minimal amount of bias, using a literature review and a small practical exercise. The results are intended to serve students studying software engineering and practitioners aiming to enhance their work practices. The three research questions above drive the entire study and continue to refine the analysis towards the key issues of cognitive bias in software development.

Literature review and thematic analysis are the methods of research that were adopted in this thesis. Scholarly and non-scholarly literature is examined and divided into themes connected with cognitive bias and processes of debiasing. Chapter 3 describes the research methodology and findings.

1.2 Structure of the Thesis

This thesis has four major chapters. The introduction is the first chapter. It provides the background

of the study and the rationale for why cognitive bias in software development is a significant issue to investigate. The aims, objectives, and research questions of the research are also mentioned in this chapter. Moreover, it also provides a brief description of how the research was conducted and executed.

Within the second chapter, there is a literature review about software development as a cognitive bias. It puts forward the notion of Cognitive bias and how various forms of bias manifest themselves when undertaking software tasks like testing, estimation, and reviewing code. The academic literature and industry sources are analysed to provide an overview of the topic and the current methods to apply in the field of debiasing.

Chapter three explains the methodology of the research of this thesis. It describes the application of thematic analysis as the primary research tool. The process of data collection is also outlined in this chapter, such as academic and non-academic sources. Moreover, it states the small bias-awareness activity conducted within the framework of the study and describes the way the data was analysed and categorized into themes.

The findings of the study are discussed in Chapter 4, which was obtained from the thematic analysis and bias awareness exercise. It is a description of the core themes that were found in the literature and how they are influenced by cognitive bias, and how they affect software development activities like testing, estimation, and decision making. The findings from the literature review and the practical activity are also discussed in the chapter, and a summary of the important findings is given at the end of the chapter.

The findings of the study are discussed and interpreted in the fifth chapter with reference to earlier academic studies and industry reports. It looks at how the findings can be applied to software development teams and how being aware of these and applying structured practices can decrease the effects of cognitive bias. The limitations of the study and recommendations for future research are also discussed.

2 Cognitive Bias in Software Development: Literature Review

This chapter is a review of the available literature and industry debate concerning cognitive bias in software development. This chapter seeks to understand the existence of cognitive bias, its manifestation in software projects, its impact on software quality, and what is now known about minimizing these impacts. Both academic and non-academic industry sources will provide a balanced perspective of both theory and practice.

2.1 Learning Cognitive Bias

This section introduces the concept of cognitive bias and explains its role in software engineering activities. Cognitive bias is a systematic aspect of cognition that affects the way individuals perceive information and make judgments. Most decisions in the case of software development are made under pressure and uncertainty. Due to this fact, developers tend to make use of mental shortcuts, which may cause biased judgments. Some of the studies clarify that cognitive bias does not mean one is unskilled but is a normal human activity (Blandford et al., 2002; Chattopadhyay et al., 2010). These biases may be reflected in software projects in estimation, testing, debugging, and teamwork.

This is also an issue that industry sources identify. According to engineering blogs and surveys, engineers tend to use the same old solutions, place too much faith in tools, or refuse to question the existing decisions (Atlassian, 2023; Martin Fowler, 2022).

2.1.1 Defining the nature of Bias

According to academic literature, cognitive bias is a disposition to give preference to some interpretation or decisions over alternatives to others when objective thinking would have dictated otherwise. According to Enoiu et al. (2022), bias is likely to manifest itself when software is being tested, whereby testers are aiming at what is already known rather than attempting to compromise the system.

Mohan and Jain (2008) explain the anchoring bias, which is the use of initial assumptions in subsequent decisions, particularly in estimation and planning. The authors of Sultana et al. (2023)

explain the problem of social bias and demonstrate that a non-technical aspect of gender or authority can affect the code review outcomes of a code. Industrially, surveys like the Stack Overflow Developer Survey (2023) indicate that developers use tools and practices that they are already familiar with, even when there are better solutions. This tendency is an indication of familiarity and confirmation bias.

2.1.2 General Biases of Software Projects

Multiple forms of bias keep recurring in the software development literature. The most common is confirmation bias. It happens when developers or testers seek ideas that confirm their original assumptions instead of trying to find mistakes (Enoiu et al., 2022).

Anchoring bias interferes with the estimation of effort and schedule. First numbers have very high power over subsequent planning decisions in situations where new information is present (Mohan and Jain, 2008). Automation bias takes place when developers use tools, test frameworks, or AI systems excessively and disregard conflicting information (Hoffmann et al., 2023; De Alwis, 2025). There are also group-related prejudices. Ahmadi and Jafari (2025) demonstrate that groupthink and authority bias may ensure that the team members do not challenge decisions. This is supported by the Project Aristotle (2023) by Google, which revealed that psychological safety is the key to team performance. Besides software development, cognitive biases are also present in governance, as explained by Jafari (2026). When policymakers or rulers make decisions based on their first assumption, without checking out new information, they can make biased decisions. Salman (2016) says that cognitive biases are a serious problem in software testing. Most studies have focused on confirmation bias; several other biases can impact testers' work. Time is a factor, and company culture can make these biases worse.

For example, overconfidence bias could cause developers to believe they have created better code than they have, leading them to not test their code as thoroughly as they should (Enoiu et al., 2022). Another sunk cost fallacy is that teams continue to invest effort in something that isn't working because they have already invested so much time in it, rather than looking at other methods. There is also the availability heuristic, which involves people making decisions based on data that comes to mind with ease, such as projects that they recently worked on, instead of all the data available (Matsubara, 2022). The tricky thing about these biases is that they can combine to reinforce one another. For instance, a team may have a first impression bias of an initial estimate, but then a confirmation bias in only seeking evidence to validate that estimate, and then a groupthink

mentality, where no one challenges the decision. The fact that knowledge of biases is not limited to the names, but also to understanding how they can interact to generate larger issues in software projects.

2.2 Impact of Biases on Software Quality

This subsection discusses the impact of cognitive and social biases on software quality and team decision-making. Cognitive bias may have a direct impact on software quality. The most common outcomes of this are missed defects, inaccurate estimations, and unfair evaluations, as discussed in the literature. The researchers demonstrate that confirmation bias in testing results in fewer bugs found by the researchers (Enoiu et al., 2022). As Sultana et al. (2023) show, bias in the process of reviewing the code may have an impact on the feedback delivery and the contributions of the individual whose contributions are appreciated. Enoiu et al. (2022) mention overconfidence bias, resulting in the omission of testing by developers.

Likewise, non-academic sources also report the same problematic issues. The article GitHub engineering blogs (2023) is about the impact of bias in the reviews of pull requests. The article in IEEE Spectrum (2024) cautions about the misuse of AI tools, which can bring about some unknown mistakes. Another area of software estimation where cognitive biases are prevalent is the estimation of functionality. In a study conducted by Matsubara (2022), the authors established that managers and clients put pressure on software professionals. They adjust estimates to accommodate others even if they are not realistic. This results in poor decision-making and poor-quality software.

Cognitive biases are strongly related to human error in safety-critical software systems. Shryane et al. (1998) investigated the types of error made by software designers in the design of railway signalling systems. They discovered that Shryane et al. (1998) studied human errors in a safety-critical software design. They found that there can be problems if they don't detect them early, even if it's a minor error. They recommend that teams try out different methods of error checking to detect more errors. These biases can impact not only small tasks but the entire project. If, for instance, development is too optimistic or it is only problems that can support the first conception that developers can find, then the important problems may not appear until later. This can make fixing bugs much more expensive and cause delays in releasing the software (Matsubara, 2022). The technical problems are not the only ones affected by bias; there's also a bias in how teams operate together.

Sultana et al, (2023) discovered that social biases in code reviews can make some members of the team feel less welcome to participate, compromising the team's ability to benefit from different viewpoints that may help identify errors. As Google's Project Aristotle (2023) explained, if people feel they are not safe to challenge assumptions or to raise potential issues, they will not do it. This can lead to a vicious circle of reinforcing bias and inability to detect quality problems. In safety-critical systems, even minor biased design decisions can result in significant human mistakes if they are not detected during design with the appropriate checking, as demonstrated by Shryane et al. (1998). Everything here demonstrates that there is more to doing something about cognitive bias than just individual effort; it's also a matter of paying attention to team and process.

2.3 Expectancies of Debiasing at this moment

According to recent studies, it is not feasible to approach bias elimination, though the mitigation of its effects can be achieved. Methods that are awareness-based are usually proposed. According to Enoiu et al. (2022), even with easy training and reflection, decision-making can be enhanced. In their article, Hoffmann et al. (2023) offer the concept of fairness audits of AI-supported development tools. According to Mohan and Jain (2008), traceability can be used to underpin objective reasoning. Awareness techniques are also supported by industry practices. According to Atlassian (2023), organised retrospectives are recommended. Martin Fowler (2022) also challenges teams to make their assumptions. Microsoft Learn blogs (2024) embrace the use of checklist-based testing as a way of preventing bias. The same idea is used in the small awareness exercise carried out in this thesis, which aims at increasing awareness to aid the improvement of attention during the testing.

Additionally, people may not act in a fully rational manner when exposed to algorithmic suggestions (Mahmud, 2024), as they may have prior assumptions, trust, and hesitation in interpreting the suggestions. This applies to software development, as developers are increasingly using AI-driven tools and other automated decision-making systems. Adepoju & Adepoju (2025) explore the issue of bias that can occur in AI systems if developers lack complete data sets or don't test the results of AI tools. They recommend that software teams periodically review their AI systems to identify and address any potential biases. This demonstrates that the need for debiasing is far from a theoretical concern. It's also applicable to normal software projects. It's important to begin with basic steps such as awareness-raising and decision review.

2.4 Debiasing Tools, Techniques, and Frames in Software Development

In this subsection, we consider the elements of actual tools and organized practices suggested in the literature aimed at attenuating the impact of cognitive and social bias in software development activities. Although the full removal of bias is deemed impractical, the past studies indicate that biased decision-making can be mitigated with the help of structured procedures and facilitating mechanisms. The application of structured checklists in testing and code review is one of the debiasing methods that has been widely discussed. Checklists help to focus on less obvious instances of failure and minimize the use of assumptions. Software testing research suggests that a checklist-based strategy leads to better fault detection and motivated searching instead of speculative searching (Enoiu et al., 2022). Practitioners in the industry also state that testing charts and predetermined heuristics of a test assist testers to overcome confirmation bias.

Another technique that has been argued in estimation studies is the reference-based estimation, in which historical data of a project is directly applied to overcome the effects of anchoring. Mohan and Jain (2008) suggest that individual anchoring can be minimized through structured estimation methods, including planning poker or reference class forecasting, because these methods encourage group thinking. These methods promote debate and reasonableness of estimates instead of using original values.

Debiasing mechanisms suggested in terms of anonymity and a structured form of feedback have been suggested in the context of code review processes. Sultana et al. (2023) confirm that the effect of anonymous code review on social bias based on authority and gender is negative. Templates for reviewing a pull request are also encouraged through industry platforms like GitHub, and these encourage a reviewer to examine the correctness, readability, and test coverage separately. Recent studies also highlight the role of software development utilizing AI-assisted tools. Automation can be effective, but Hoffmann et al. (2023) caution that too much automation of recommendations offers the risk of automation bias. To overcome this, fairness audits and human-in-the-loop have been proposed, under which automatically generated outputs are not accepted blindly, but are subject to questioning.

The general findings of the literature show that successful debiasing in software development does not presuppose sophisticated technology interventions. Rather, structured workflows, explicit reflection, and awareness-based tools seem to be the most viable and most endorsed.

3 Research Methodology

This chapter describes the methodology of the research done in this thesis. It explains the methodology of the research, the data used, and the analysis. There is also a description of the bias-awareness exercise that occurred in the chapter because of the mid-term thesis presentation. The primary objective of the methodology is to address cognitive bias in software creation and to discuss the easiest methods to minimize its adverse effects. The study is based on the current literature and minor practical experiments instead of large-scale experiments.

3.1 Research Design

Thematic analysis is the method of research that was employed in this thesis. A qualitative research approach is known as thematic analysis, which is applied to identify, analyse, and arrange patterns or themes in data. It is an appropriate methodology in this thesis since the purpose is to research the recurrent concepts regarding cognitive bias among various sources. The study is research literature-based and exploratory. It uses both academic research papers and non-academic industry sources. This assists in integrating scientific knowledge with practice related to real software development.

3.2 Choice of Thematic Analysis

Thematic analysis was selected since cognitive bias is a human-related subject that can be observed in various situations, including testing, estimation, code review, and teamwork. Other works in the past have written on bias in varied terms, and through thematic analysis, this is what brings these terms together. The other reason why thematic analysis has been selected is due to its flexibility. It enables the examination of academic publications and other industry publications, including blogs, surveys, and reports. This helps in achieving the objective of knowing the manifestation of cognitive bias in theory and practice.

Thematic analysis is also a suitable method for this study because of its ability to integrate the various types of data in this study which is related to cognitive bias in software development. There are some experiments (Enoiu et al., 2022) and questionnaires (Salman, 2016) that have already been used to study some parts of cognitive bias but the understanding of how cognitive bias manifests in

practice is mostly based on qualitative observation, case studies, and practitioner reflections (Atlassian, 2023; Fowler, 2019). Thematic analysis is a flexible yet systematic way of synthesizing the above-mentioned data sources, which can help the researcher to recognise patterns that transcend methodological boundaries. This corresponds to the recommendations of Blandford et al. (2002) for using qualitative methods in human-computer interaction research to account for the subtleties and context-dependent aspects of human behaviour. Furthermore, thematic analysis was well-suited to the aim of this thesis because it allows the exploration of a set of biases and debiasing strategies rather than testing specific hypotheses (Braun & Clarke, 2006). The methodology employed in the study will allow it to offer a comprehensive overview and draw together the empirical results of academic research and industry sources, resulting in a more holistic understanding of the phenomenon studied.

3.3 Data Collection Process

The information that is utilized in this thesis is academic and non-academic data on cognitive bias in software development. The academic sources were searched via LUT Primo, eLUT Library, and Google Scholar. Such sources involve peer-reviewed journal articles and conference papers that deal with software testing, decision-making, and human factors. Non-academic materials are industrial blogs, developer surveys, and engineering reports. These sources were chosen because they outline actual experiences and practices of software development teams.

3.3.1 Selection of Sources

The sources were chosen on the following criteria:

- Cognitive bias or software development relevance.
- Emphasize testing, estimation, code review, or teamwork.
- Authority of the mentioned author or organization.
- Obvious affiliation with real-life software development.

A total of 25 sources were selected for analysis, comprising 11 peer-reviewed academic articles (10 journal articles and 1 conference paper), 3 theses or dissertations, and 11 industry sources (blogs, reports, surveys, and magazine articles).

List of Included Studies

- Peer-reviewed journal articles (10): Adepoju & Adepoju (2025), Ahmadi & Jafari (2025), Blandford et al. (2002), Chattopadhyay et al. (2022), Enoiu et al. (2022), Hoffmann et al. (2023), Mohan & Jain (2008), Shryane et al. (1998), Sultana et al. (2023), Braun & Clarke (2006).
- Peer-reviewed conference paper (1): Salman (2016).
- Theses and dissertations (3): De Alwis (2025), Mahmud (2024), Matsubara (2022).
- Industry sources (11): Atlassian (2023), Fowler (2019), GitHub (2023), GitLab (2020), Google (2023), IEEE Spectrum (2024), InfoQ (2024), Jafari (2026), Microsoft Learn (2024), Stack Overflow (2023), Thoughtworks (2024).

The sources were then collected and read. Important concepts concerning the aspect of cognitive bias were mentioned and coded. Like codes were then combined to make general themes. Examples of codes include:

- Confirmation bias in testing: Confirmation bias is a bias in testing where positive results are more likely to be accepted than negative results.
- Anchoring in estimation: Anchoring bias is when a first idea has a powerful effect on subsequent decisions. For software projects, the developer may still use an early estimate even if there is more information to indicate that the original estimate is not correct.
- Authority bias in teams: The tendency to listen to suggestions from experienced or senior team members without adequate evaluation is referred to as authority bias. This can result in less discussion and fewer perspectives in decision-making.
- Awareness-based debiasing: Awareness-based debiasing is the process of raising awareness of cognitive biases to make developers aware of their own assumptions. Even the simplest awareness activities have been shown to promote more balanced decision-making and testing behaviour.

3.4 Awareness Exercise Procedure

A small awareness exercise was conducted and used as part of this thesis to explore the impact of

knowledge of these cognitive biases on software testing behaviour. The exercise was not an experiment but a practical activity to help implement the results of the literature review. A small Python program was created with two functions: grade calculation and login validation. These functions were chosen as they are easily understood and could be tested with valid and invalid inputs. In the first part of the program, the program was tested without focusing on cognitive bias. Most of the testing involved normal inputs. Once initial testing was done, the literature was consulted for information concerning confirmation bias in software testing.

The second stage involved another test of the same program. This time, greater focus was placed on unusual inputs, invalid values, and alternate scenarios. Additional test cases were conducted to investigate if different results would be found. Only the author of this thesis participated in the exercise. Therefore, the results cannot be generalized. The activity, however, gave a practical example that supports the results obtained in the literature on the topic of awareness and software testing behaviours.

3.5 Considerations of Reliability, Validity, and Research Quality

The validity and reliability of this study were taken into consideration during the process of research. Since this thesis is based on a qualitative thematic analysis, the specific focus was on data collection and analysis transparency. The reliability was facilitated by using established academic materials that were accessed in established research databases. Coding was done systematically and was performed through the occurrence of similar concepts in various sources and grouping them into themes. Although the coding was done by one researcher, consistency was ensured by going through previous codes when analysing new data.

Validity was also ensured by comparing the academic results with non-academic industry reports. This triangulation contributed to the assurance that the discovered themes are based on theoretical concepts and practical application. The variety of types of sources makes the construct validity stronger since it proves that the biases observed are not specific to the academic setting.

4 Findings

The thematic analysis led to a few repeating themes associated with cognitive bias in software development. A lot of the literature indicates that cognitive bias can be minimized most effectively by means of awareness. The scholarly research indicates that even brief descriptions of prejudice may enhance attention when testing. It is also advised by industry sources that training sessions, as well as checklists and reflection practices, should be employed to aid the improvement of decision-making. This theme confirms the fact that developers do not require sophisticated tools to begin eliminating bias. Basic awareness can alter the world.

4.1 Theme 1: Awareness and Training Methods

One important theme that can be observed in the literature is that awareness is critical in minimizing cognitive bias during software development. Studies indicate that developers and testers are usually not aware of biases such as confirmation bias when testing or overconfidence in their decisions (Enoiu et al., 2022). The simplest forms of awareness interventions, such as short-term training, contemplation, and bias clarification, can contribute to individuals being more careful and critical at work.

The sources in the industry support this observation and suggest such practical steps as retrospectives, testing checklists, and reminders when doing a review (Atlassian, 2023; Microsoft Learn, 2024). The mini-bias-awareness test that was performed in the present thesis also concurs with these results because the attention during the test increased upon learning about cognitive bias. In general, the literature indicates that the use of awareness-based strategies is an effective and low-effort method of assisting in the enhancement of decision-making.

4.2 Small Bias-Awareness Testing Exercise

A small experiment was performed to see if knowledge of cognitive bias would affect software testing behaviour. A basic program in Python was written that has two functions, one to calculate a discount and another to check the user's login (Figure 1). This program was piloted in two phases. The testing in the first stage was applied to normal inputs.

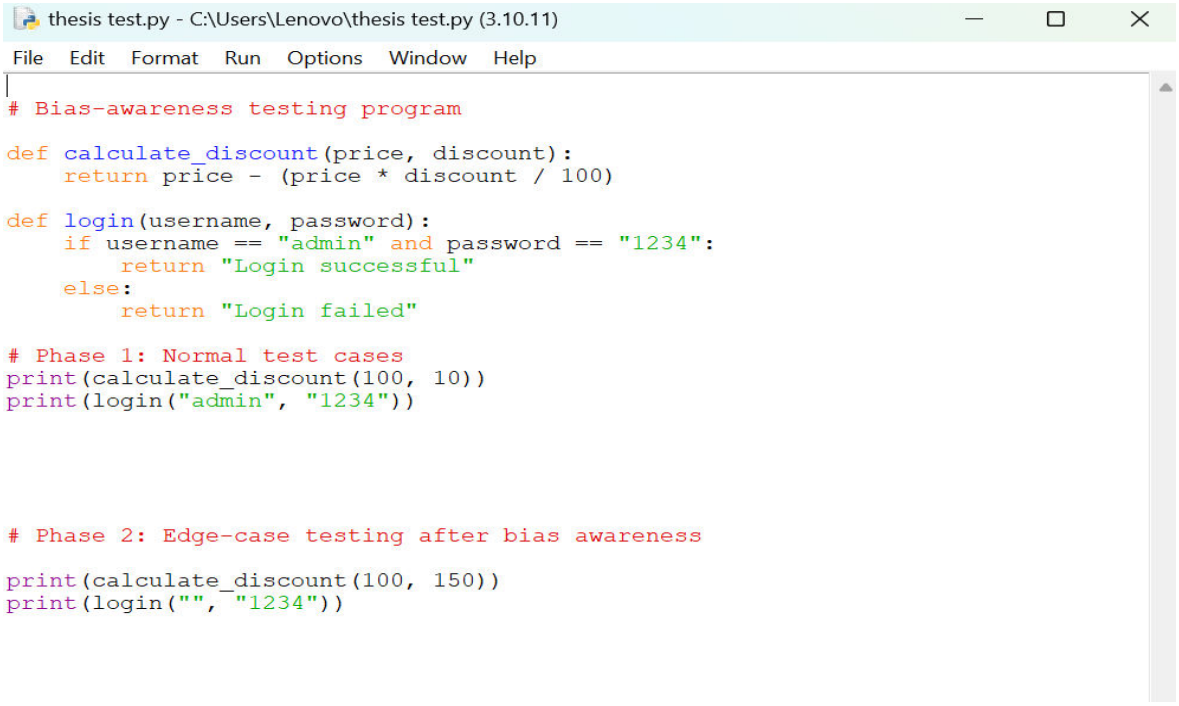

Figure 1. Python testing program used in the bias-awareness exercise.

In the second stage, the same program was restarted, and the concept of confirmation bias and looking at the edge cases and alternative scenarios were considered. The exercise did not aim to produce generalisable results, but rather to give a small example of what might happen if one is aware of bias in testing.

4.2.1 First Testing Phase

During the first phase of the exercise, the program was tested using expected and normal inputs. This was done to observe what happens when the program is used normally. The testing focused mainly on successful outputs rather than unusual inputs or failure situations. Figure 2 shows the result of the first testing phase

Figure 2. Output from the first testing phase before bias awareness.

Figure 2 shows the result of the first testing phase. The emphasis of the testing during this phase was on the normal behaviours of the program and the expected results. The login function was checked using valid credentials, and the calculation part was tested with normal positive values. The program seems to have performed as expected according to these tests. This part didn't involve much consideration of edge cases or failure scenarios, though. This is like the concept of confirmation bias in software testing, which means that testers tend to concentrate on potential positive outcomes rather than anomalies that could uncover potential latent issues (Enoiu et al., 2022; Chattopadhyay et al., 2010).

4.2.2 Second Testing Phase

The same program was tested again after an examination of confirmation bias and the need to test alternative cases. The second phase was different from the first, as the emphasis shifted from just normal to unusual input, failed logins, and edge cases, which were not covered in the first phase.

The second phase of the exercise, following the introduction of bias awareness, is shown in Figure 3.

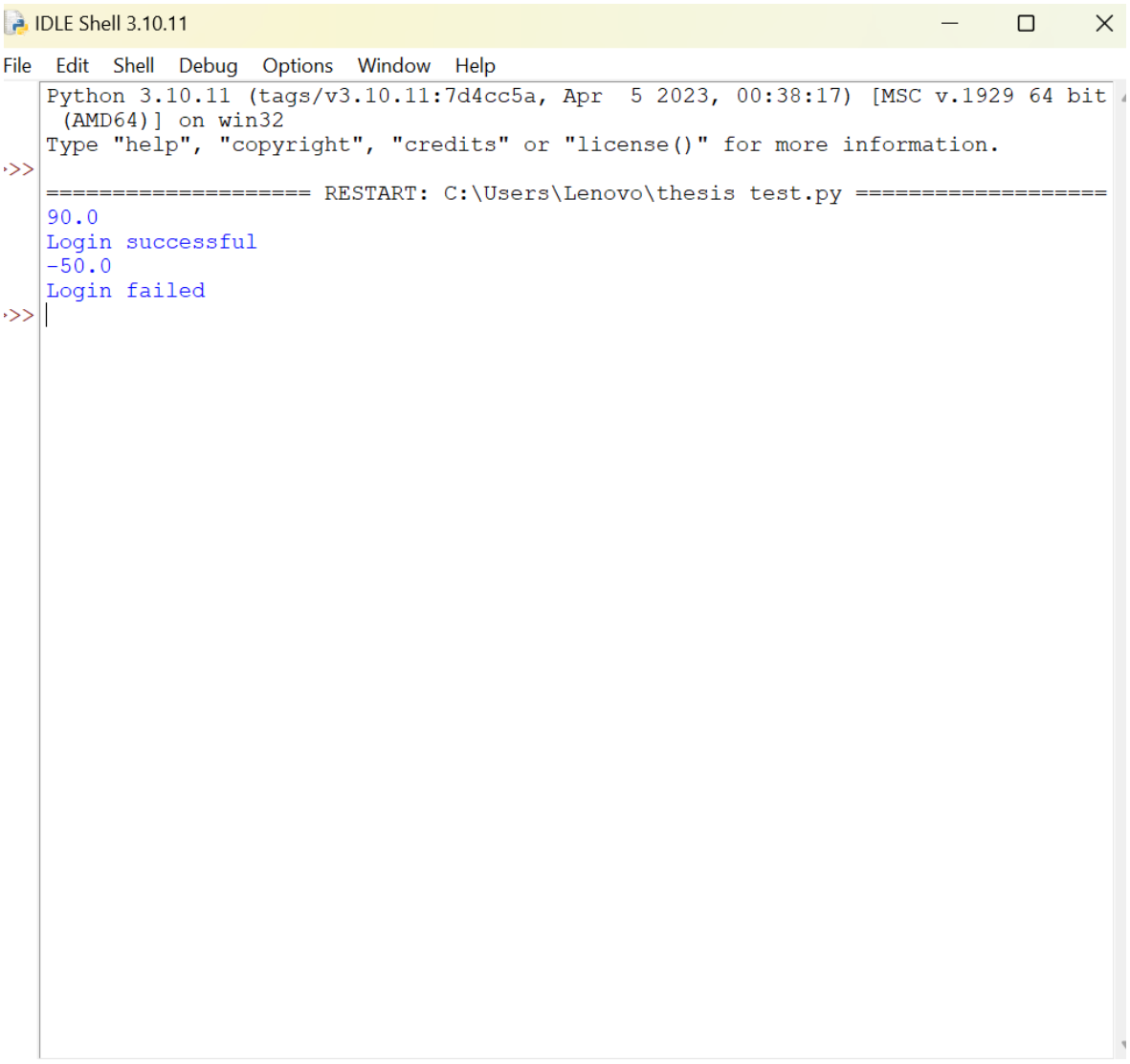

Figure 3. Output of the second testing phase.

The test during this phase focused on less predictable inputs, and more emphasis was placed on the alternate cases, which included mis-entered login data and nonstandard data inputs for calculation. The latter testing approach was more extensive and more fundamental. The second phase identified behaviours that were not observed in the first phase. This aligns with previous studies that indicated that being aware of confirmation bias may increase testers' inclination to investigate unexpected results and focus on potential defects and "edge cases" (Enoiu et al., 2022).

4.2.3 Visual comparison of the exercise

The observations from the exercise were summarised visually, and the difference between testing phases was highlighted. The first chart shows the differences in the number of issues/weaknesses identified in terms of bias awareness before and after bias awareness (Figure 4).

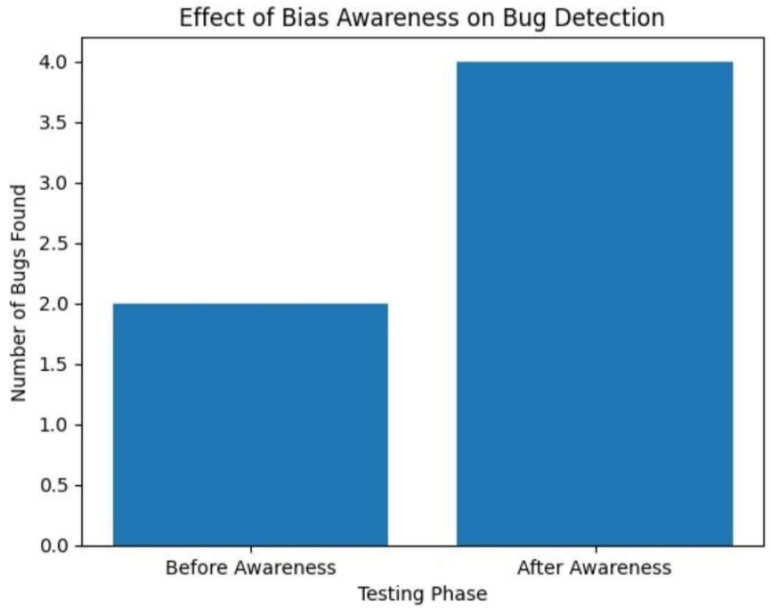

Figure 4. Number of issues identified before and after Bias Awareness.

As shown in Figure 4, the number of bugs detected increased after bias awareness. The figure helps to confirm the assumption that has been made in previous studies that awareness can be used to influence decision-making without elaborate tools and extensive training.

The second visual summary is based not on the number of issues found, but on how testing behaviour changed. It demonstrates the shift in the testing focus from what is expected to happen to a broader and more critical look at software.

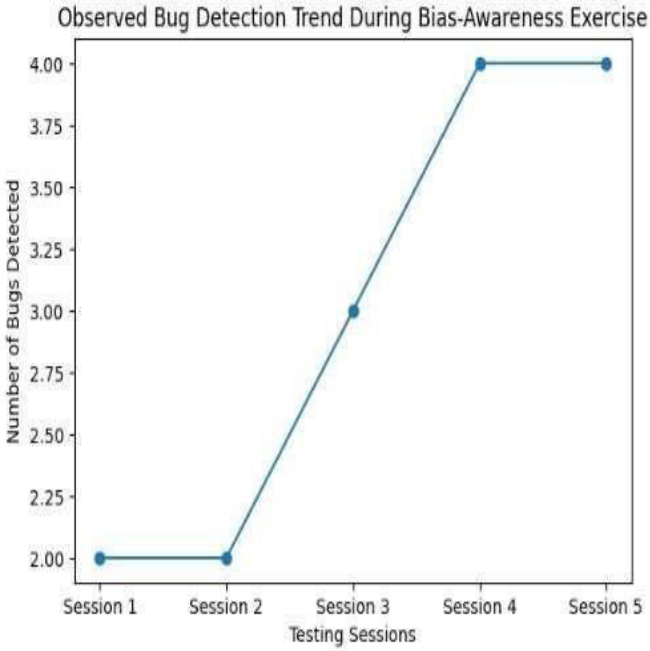

Figure 5. Change in testing focus before and after bias awareness

Figure 5 shows the change in testing behaviour in the exercise. Initially, the focus of the testing was primarily on normal program operation and successful output. They discussed confirmation bias after that and turned the discussion to other inputs, failures, or weaknesses. This is important because it demonstrates the impact that an awareness activity, even a brief one, can have on the approach to testing. In the context of this thesis, the figure supports that awareness-based debiasing may contribute to a more balanced and critical testing process.

4.3 Theme 2: Team and Process Practices

The other important theme is team practices and organized procedures that aid in curbing prejudice in group decision-making. According to academic literature, activities such as estimation and code review are also subject to authority bias, groupthink, and conformity (Ahmadi and Jafari, 2025; Sultana et al., 2023). Planning poker, anonymous reviews, and review templates are the structured approaches that promote more balanced conversations and decrease the use of assumptions. The same can be seen in industry reports, where psychological safety and the ability to understand the team processes were highlighted (Google, 2023; GitHub, 2023). In situations where teams have systematic practices and feel free to challenge decisions, the chances of biased judgments do not greatly impact the outcomes. These results also show that debiasing is not merely a personal exercise but requires a conducive team culture and development.

4.4 Summary of Findings

There were two main themes identified with the thematic analysis of the cognitive bias in software development. The first theme was on awareness and training approaches, and the second theme was on team practices and structured development processes. Academic and industry literature have repeatedly stated that cognitive bias has the potential to have a detrimental impact on testing, estimation, and decision-making processes. The results suggest that awareness-based interventions can enhance attention in software tasks. Simple interventions like reminders, reflective exercises, and testing checklists were even proven to contribute to more careful thinking and to the elimination of assumptions. This was confirmed by a small testing activity that was carried out as part of this thesis, which tested some more edge cases that were discovered after learning about confirmation bias.

Communication between the team and a structured workflow were the second theme. The literature suggested various practices to minimize authority bias and groupthink, including anonymous reviews, planning poker, and open discussion. It moves away from a subjective assessment by the developer and allows them to consider their decisions based on evidence and not hierarchy. The overall results indicate that cognitive bias is a widespread phenomenon in software development, but its negative impact can be mitigated by being aware of it, collaborating, and adopting simple, structured practices. The themes of this study also provide an answer to the research questions presented in the introduction chapter.

4.4.1 Summary of Theme 1

Awareness and training methods were the first theme found in this study. According to both academic and non-academic sources, it is challenging to realize cognitive bias because of the unconsciousness that happens during software development activities. It has been argued that developers and testers might only think about what they expect and ignore the possible other outcomes, especially when making decisions and testing (Enoiu et al., 2022).

Literature showed that awareness-based approaches can help in reducing these effects. Short training sessions, reflection activities, test checklists, and conversations about cognitive bias were often suggested as simple techniques (Atlassian, 2023; Microsoft Learn, 2024). This result was confirmed by the small-scale exercise done in this thesis. Before this, after learning about

confirmation bias, more test cases and edge cases were tried, and it was found that there were more problems that were not initially thought of. Overall, the results indicate that awareness is a very real and attainable means of enhancing decision making and software testing quality.

4.4.2 Summary of Theme 2

The second theme was centred around team and process practices to minimize cognitive and social bias in software development. The literature reviewed revealed that biases do not only exist on an individual level but can also manifest in team discussions, estimation activities, and even code reviews. In software teams, some common problems that can impact decision quality were identified, such as authority bias, conformity, and groupthink (Ahmadi & Jafari, 2025; Sultana et al., 2023).

Academic research and industry reports both highlighted the need for processes that are structured to minimize these risks. Planning poker, review templates, testing checklists, and open team discussions were often recommended, as these practices help to avoid the making of assumptions in favour of objective evaluation (Google, 2023; GitHub, 2023). The results indicate that debiasing is not solely a personal or individual issue, but also a team issue and process. A balance of awareness and structured practices will improve the likelihood of balanced and evidence-based decision-making.

4.4.3 Relationship Between Themes

The themes identified in this study are closely related. Awareness and training techniques enable members to identify the potential for bias in their thinking, and team and process techniques offer structured methods to reduce the impact of bias on software development activities. There is literature on the fact that even awareness is not always enough. Developers can be aware of cognitive bias and make cognitive-biased decisions when time is limited or uncertainty is present. Structured practices like code review templates, planning poker, and testing checklists can offer more support for this reason. Similarly, team practices are more effective when team members know about common biases. Teams can make more balanced decisions by having open discussions, questioning assumptions, and asking for other perspectives.

In general, results suggest that the solutions to decrease cognitive bias are not complicated, and also not costly. Most of the methods that have been found in the literature are simple and can be integrated into daily software development. This can be in the form of structured testing checklists, review templates, planning poker, open discussion, frequent reflection on decisions, etc. Together with knowledge of some common cognitive heuristics, software teams can be better equipped to spot assumptions, consider alternative options, and make more informed decisions (GitHub, 2023; Google, 2023; Thoughtworks, 2024; Mahmud, 2024).

The three parts of the work, in addition to the practical exercise (thematic analysis and the literature review), confirm the same result. While cognitive bias is inherent in human decision-making, its detrimental impact on software development can be mitigated by being aware of it, developing processes that structure bias, and fostering a team culture that fosters collaboration. This is in line with both the academic research and industry recommendations presented in this thesis (Enoiu et al., 2022; Atlassian, 2023). These research findings give a smooth transition to the discussion in the next chapter and help to answer the research questions raised at the start of this thesis.

5 Discussion

The chapter is based on the results of the thesis concerning its connection with previous scholarly studies and the real-life experiences in the industry. It is discussed on the basis of the literature review, thematic analysis, and the small bias-awareness exercise in the mid-term thesis presentation. The interpretation of the results is based on the comparison with older studies and the reflection on a practical approach in software development. Lastly, the weaknesses of the research are discussed.

5.1 Interpretation of Findings

The findings of this thesis indicate that cognitive and social biases are widespread in most software development activities. The need to work under pressure or uncertainty has already been identified by academic studies as making developers and testers become reliant on mental shortcuts. As an example, prior studies denote the importance of confirmation bias in software testing since they indicate that testers tend to concentrate on positive results rather than actively scrutinize the software (Enoiu et al., 2022; Chattopadhyay et al., 2010).

The thematic analysis assisted in clustering similar ideas identified in academic papers and industry sources. One of the themes was repeated, which was that bias is frequently unconscious. This manifested in academic publications and non-academic ones, such as blogs and surveys, where developers talk about using what they know best or initial assumptions (Mohan and Jain, 2008; Atlassian, 2023; Martin Fowler, 2022).

These findings can be justified by the small bias-awareness exercise undertaken in this thesis. Before being informed about confirmation bias, the testing was focused primarily on normal cases. Following the reading of a brief description of the bias, further tests of edge cases were conducted, and more faults were discovered. The same has been reported in the academic literature, where even brief training on awareness-enhanced testing behaviour (Enoiu et al., 2022).

These remarks were also contained in industry sources. Decisions about software often rely on past experiences and assumptions and may thus lead to additional cognitive bias (Atlassian, 2023). This reinforces the results of this thesis, which suggest that such awareness activities have the potential to make developers more critical.

5.2 Comparison to the preceding Reports

The findings are next compared with the previous reports provided below. During the comparison of findings of this thesis with past scholarly research, it is possible to observe strong similarities. Cognitive bias studies on software engineering show that confirmation bias, anchoring bias, and automation bias recurrently occur during the testing and estimation process (Mohan and Jain, 2008; Enoiu et al., 2022; Hoffmann et al., 2023).

Other social biases, including the authority bias and groupthink, have also been discussed in recent scholarly articles, particularly in a team setup (Ahmadi and Jafari, 2025; Sultana et al., 2023). These results also fit other non-academic industry reports, like the Project Aristotle by Google (2023), in which the authors highlight the value of psychological safety and open team discussion. Moreover, non-academic websites such as Stack Overflow surveys (2023) indicate that developers tend to use what they know and stick to it, which is anchoring and selection bias. The information in the findings of this thesis is more reliable because academic research has been consistent with the observations in the industry. The results also confirmed industry reports. Google (2023) pointed to psychological safety and open communication in teams, and Atlassian (2023) to awareness in mitigating the effects of bias on decision-making. The conclusions drawn from academic research, which are the subject of this thesis, are corroborated by these observations.

5.3 Practical Implications on the Software Development of Teams

Simple practices can be adopted by software development teams that will decrease cognitive and social bias in the day-to-day development process. This thesis's research literature review indicates that a large portion of software problems are not solely technical but also result from human decision-making. Developers and testers are frequently subjected to deadlines and uncertainty, leading to the possibility of using assumptions and mental shortcuts (Enoiu et al., 2022). One of the implications in practice is the use of awareness-based activities. Devil's Advocate reminders, short training sessions, or team discussions can help developers consider their choices with a critical mind. Even a simple reminder of the possibility of confirmation bias has been shown to prompt testers to consider alternative scenarios besides the anticipated one, rather than only looking for those (Enoiu et al., 2022). Recommendations are also found in industry sources to reflect and continuously learn in the development team (Atlassian, 2023).

One of the other implications is the use of structured development practices. Testing checklists, planning poker, and code review templates can help with more objective decision-making, minimising the personal assumptions made. An evidence and requirement-oriented approach help teams to focus on the facts and requirements, not just on intuition (Mohan & Jain, 2008; Sultana et al., 2023). Structured review processes and collaborative decision-making have also been reported to be positive experiences in the industry (GitHub, 2023; Microsoft Learn, 2024).

The results also point to the significance of team culture. Everyone on the team should be comfortable giving other team members their own ideas and doubts when they are needed. Ahmadi & Jafari (2025) and Sultana et al. (2023) conducted research on software teams where they found that members of software teams may feel reluctant to challenge the status quo, which creates an environment conducive to authority bias and groupthink. Psychological Safety is also a key ingredient in Google's Project Aristotle teamwork framework (Google, 2023). Therefore, emphasis on open communication can facilitate the group's making better-informed and balanced decisions. As AI-driven development tools become more prevalent, software teams are also facing new challenges. These tools can help the developer be more productive, but they can also make the developer too reliant on the suggestions provided and may not check the output thoroughly. This has been shown in previous research as automation bias and suggests that human oversight is recommended when using AI tools (Hoffmann et al., 2023; De Alwis, 2025). Therefore, AI-generated outputs should be considered a tool for teams to use and not a definitive answer.

The results collectively indicate that decreasing cognitive bias doesn't need to be expensive or complicated. Software development teams (SDTs) can benefit from software awareness activities, such as simple awareness activities, processes, and environments, which can impact software decision-making, software quality, and collaboration (Atlassian, 2023). Well-designed review processes can also help to achieve more impartial decision-making. According to GitHub (2023), structured feedback processes make sure that code reviews are carried out with objectivity, since they help the evaluator concentrate on the evidence instead of the assumptions. Likewise, Microsoft Learn (2024) suggests using checklist test approaches to minimize human error and enhance uniformity in software testing pursuits.

There are also indications from industry sources that simply making team software as diverse as possible is not enough to overcome bias – it also requires practices to be established. When software is developed across individuals and tasks, it is vital to communication, collaboration, and shared review practices (GitLab, 2020). Similarly, Thoughtworks (2024) points to the importance of using sensible defaults, shared practices, and reflective discussion to make technical decisions

rather than making decisions in isolation. Testing quality can also be improved by implementing more effective test architectures and test practices (InfoQ 2024), which can enable teams to pay more attention to outcomes beyond the ones they expected. These sources all indicate that bias reduction in software development is enhanced by raising awareness levels and by adopting a structured software team approach, reporting structure, and intentional testing procedures.

5.4 Limitations of the Study

This research study has several constraints. To begin with, the bias-awareness exercise only included one subject, which restricts the extrapolation of findings. Second, the study is primarily based on a literature review and a small practical experiment as opposed to research on a large scale. Moreover, there were non-academic materials, including blogs and surveys, to mark the actual practice. Though these sources can be very helpful to see how the industry works, they are not always in accordance with the principles of strict scientific research work. The research may be expanded in the future by using larger samples, longer observation periods, or controlled experiments to determine the effects of bias awareness on software development teams over a longer period.

5.5 Future Research

This thesis was based on a literature review and a brief awareness-raising exercise. Further research can be conducted with larger software development teams and in real project settings. A potential approach is to carry out controlled experiments with several developers or testers. These studies may be used to determine if there are any significant improvements in software quality, defect detection, and/or software estimation accuracy following awareness training.

An alternative would be to explore the effect that cognitive bias has on AI-assisted development tools. The growing adoption by developers of tools like code generation assistants will make it even more critical to understand automation bias. Other potential areas for investigation in the future are cognitive bias in other software engineering tasks, such as Requirements Engineering, Project Management, and System Design. These areas are subject to key decisions, and the judgment of the human being may also play a role.

6 Conclusion

This thesis explored the subject of cognitive and social biases in software development and included references to the impact of such processes on software testing and related choices. The paper has incorporated a structured literature review, thematic analysis, and a small practical study with awareness as a form of activity to cover both theoretical and practical observations. The issue was chosen due to the large number of problems in software projects that are caused by human cognition and not technical factors. Research on cognitive bias assists the researcher in explaining how defects can go unnoticed, estimates can be inaccurate, and decisions can seem sound, yet yield poor results. The findings indicate that cognitive biases, especially confirmation bias, are prevalent in software development activities. It is always documented in the literature that developers are more likely to seek evidence that will uphold their original assumptions. The same was also the case in the awareness exercise, where testing initially concentrated on expected behaviours. The results indicate that awareness of bias can positively affect test behaviours and decision-making. The findings address the research questions and conclude that cognitive biases are real in software development, they impact the quality of software, and that the negative impact can be lowered with the help of merely awareness-based strategies. The results of this thesis may be implemented in education and in real-world software development. The future work and recommendations are as follows:

- Using bias-awareness exercises on bigger and more representative populations of developers and testers.
- Research on long-term bias awareness consequences on software quality and collaboration.
- Incorporating topics of cognitive bias into programs of software engineering education and training.
- The expansion of the research to other stages of development, like estimation, requirements analysis, and code reviews.

References

Adepoju, S. & Adepoju, M. (2025). Cognitive biases: Understanding and designing fair AI systems for software development. International Journal of Advanced Computer Science and Applications, 16(9). Saatavissa DOI: 10.14569/IJACSA.2025.0160901.

Ahmadi, K. & Jafari, S. A. S. (2025). A realistic review of cognitive biases in governance decisions. *Strategic Discourse Quarterly*, 2(5), 47-72.

Atlassian. (2023). Understanding cognitive bias in agile teams. Atlassian Engineering Blog. Saatavissa: https://www.atlassian.com/ (Accessed 24.6.2026).

Blandford, A., Furniss, D. & Makri, S. (2002). Qualitative HCI research: Going behind the scenes. *Human–Computer Interaction*, 17(2-3), 259-282. Saatavissa DOI: 10.1207/S15327051HCI1702_3.

Chattopadhyay, S., Nelson, N., Au, A., Morales, N., Sanchez, C., Pandita, R., & Sarma, A. (2022). Cognitive biases in software development. *Communications of the ACM*, 65(4), 115-122. Saatavissa DOI: 10.1145/3517217.

De Alwis, R. D. (2025). *The comparison of confirmation bias between human software testers and ChatGPT*. Master's thesis. Lappeenranta-Lahti University of Technology LUT, School of Engineering Sciences. Saatavissa: https://lutpub.lut.fi/handle/10024/169769 (Accessed 24.6.2026).

Enoiu, E. P., Cusmaru, A. & Malm, J. (2022). Unveiling cognitive biases in software testing: Insights from a survey and controlled experiment. *Information and Software Technology*, 145, 106809. Saatavissa DOI: 10.1016/j.infsof.2022.106809.

Fowler, M. (2019). Is high-quality software worth the cost? *Martin Fowler*. Saatavissa: https://martinfowler.com/ (Accessed 10.6.2026).

GitHub. (2023). Improving pull request reviews through structured feedback. GitHub Engineering Blog. Saatavissa: https://github.blog/ (Accessed 10.6.2026).

GitLab. (2020). Improving DevOps and software development with communication and collaboration. Saatavissa: https://about.gitlab.com/blog/collaboration-communication-best-practices/ (Accessed 21.6.2026).

Google. (2023). Project Aristotle: Understanding effective teams. Google re:Work. Saatavissa: https://rework.withgoogle.com/ (Accessed 10.6.2026).

Hoffmann, A. L., Kleinberg, J., & Mullainathan, S. (2023). Fairness audits for AI systems in software development. *Communications of the ACM*, 66(11), 62-69. Saatavissa DOI: 10.1145/3593012.

IEEE Spectrum. (2024). The hidden risks of AI-assisted software development. *IEEE Spectrum*. Saatavissa: https://spectrum.ieee.org/ (Accessed 10.6.2026).

InfoQ. (2024). Reaching your automatic testing goals by enhancing your test architecture. Saatavissa: https://www.infoq.com/testing/ (Accessed 21.6.2026).

Jafari, A. S. (2026). A realistic review of cognitive biases in governance. Department of Islamic Philosophy and Theology, University of Qom.

Mahmud, H. (2024). *Algorithms in decision-making: exploration of algorithm aversion and appreciation*. Väitöskirja. Lappeenranta-Lahti University of Technology LUT, School of Engineering Science. Saatavissa URN:ISBN:978-952-412-153-8.

Matsubara, P. G. F. (2022). *From a gambit to a defense with SwEDeL: an approach to defend software estimates from pressure*—doctoral dissertation. Universidade Federal do Amazonas, Brazil.

Microsoft Learn. (2024). Checklist-based testing to reduce human error. Microsoft Learn. Saatavissa: https://learn.microsoft.com/ (Accessed 10.6.2026).

Mohan, K. & Jain, R. (2008). Using traceability to mitigate cognitive biases in software development. *Communications of the ACM*, 51(9), 66-72. Saatavissa DOI: 10.1145/1378727.1389970.

Salman, I. (2016). Cognitive biases in software quality and testing. *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion (ICSE-C)*, 823-826. Saatavissa DOI: 10.1145/2889160.2889265.

Shryane, N. M., Westerman, S. J., Crawshaw, C. M., Hockey, G. R. J. & Sauer, J. (1998). Task analysis for the investigation of human error in safety-critical software design: a convergent methods approach. *Ergonomics*, 41(11), 1719-1736. Saatavissa DOI: 10.1080/001401398186153.

Stack Overflow. (2023). Developer Survey 2023. Stack Overflow Insights. Saatavissa: https://insights.stackoverflow.com/ (Accessed 10.6.2026).

Sultana, S., Turzo, A. K. & Bosu, A. (2023). Code reviews in open-source projects: How do gender biases affect participation and outcomes? Saatavissa: https://arxiv.org/abs/2210.00139 (Accessed 10.6.2026).

Thoughtworks. (2024). Better technology decision-making is underpinned by 'sensible defaults'. Saatavissa: https://www.thoughtworks.com/en-cn/insights/blog/technology-strategy/better-tech-decision-making-underpinned-by-sensible-defaults (Accessed 10.6.2026).

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77-101. Saatavissa DOI: 10.1191/1478088706qp063oa.