Iflaah Salman

THE EFFECTS OF CONFIRMATION BIAS AND TIME PRESSURE IN SOFTWARE TESTING
IFLAH SALMAN

THE EFFECTS OF CONFIRMATION BIASES AND TIME PRESSURE IN SOFTWARE TESTING

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Information Technology and Electrical Engineering of the University of Oulu for public defence in the OP-Pohjola auditorium (L6), Linnanmaa, on 11 December 2019, at 12 noon

UNIVERSITY OF OULU, OULU 2019
Salman, Iflaah, The effects of confirmation bias and time pressure in software testing.
University of Oulu Graduate School; University of Oulu, Faculty of Information Technology and Electrical Engineering
*Acta Univ. Oul. A 735, 2019*
University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

**Abstract**

**Background:** Confirmation bias is the tendency to search for evidence that confirms a person's preconceptions. Confirmation bias among software testers is their tendency to validate the correct functioning of the program rather than testing it to reveal errors. Psychology literature suggests that time pressure may promote confirmation bias because time pressure impedes analytical processing of the task at hand. Time pressure is perceived negatively for its effects in software engineering (SE), therefore, its effect on confirmation bias may exacerbate software quality.

**Objective:** We aim to examine confirmation bias among software testers. Additionally, we examine the effect of time pressure on confirmation bias and how time pressure affects the testers' perception of the performance. We also question what other antecedents to confirmation bias exist in software testing and how they lead to it.

**Method:** We first examined the state of the art research on cognitive biases in SE using systematic mapping. Then, we empirically examined the feasibility of using students in further experiments. An experiment with 42 students (novice professionals) investigated the manifestation of confirmation bias and whether time pressure promotes it. Another experiment with 87 novice professionals examined the perception of the performance of software testers under time pressure. A grounded theory study based on the interview-data of 12 practitioners explored other antecedents to confirmation bias in software testing and how they lead to it.

**Results:** Time pressure emerged as a major antecedent to confirmation bias in the grounded theory. Testers prefer to validate the correct functioning of the program under time pressure. However, time pressure could not significantly promote confirmation bias among testers. Software testers significantly manifest confirmation bias irrespective of time pressure. The perception of performance is also sustained irrespective of time pressure.

**Conclusion:** Testers should develop self-awareness of confirmation bias and improve their perception of performance to improve their actual testing. In the industry, automated testing may alleviate confirmation bias due to time pressure by rapidly executing the test suites.

**Keywords:** cognitive bias, confirmation bias, controlled experiment, grounded theory, interview, software engineering, software testing, systematic mapping, time pressure
Salman, Iııah, Vahvistusharhan ja aikataulupaineen vaikutukset ohjelmistojen testauksessa.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Tieto- ja sähköteknikan tiedekunta
Acta Univ. Oul. A 735, 2019
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä


Tavoite: Tarkastelimme ohjelmistotestaajien vahvistusharhaa tutkimalla aikataulupaineen vaikutusta vahvistusharhaan ja testaajien käsitykseen testauksen tehokkuudesta. Lisäksi kysymme, mitä muut tekijät johtavat ohjelmistotestauksen vahvistusharhaan, ja millä tavoin.


Johtopäätös: Ohjelmistotestaajien on syystä kehitettä tietoisuutta vahvistusharhasta ja parantaa käsittelyään työn tehokkuudesta parantakaan testaustyötä. Teollisuudessa automaattinen testaus on lienevän aikataulupainesa aiheuttamaa vahvistusharhaa nopeuttamaan testisarjoja, erityisesti aikataulupaineen vaikutus testausten tehokkuuteen.

Asiakirjat: aikataulupaine, ankkuroidut teoria, haastattelu, kognitiivinen harha, ohjelmistotekniikka, ohjelmistotestaus, systemaattinen kirjallisuuskartoitus, vahvistusharha, valvottu koe
To my late father, Sllman.
A lead role alone cannot make a successful theatrical performance without a remarkable performance by other cast and crew. To me, similar is the case with earning a PhD where multiple roles contribute to the whole performance.

My foremost thanks go to my supervisor, Prof. Burak Turhan, who has always been more optimistic than me. Thanks a lot to Dr Pilar Rodriguez, my co-supervisor, who always believed in me and has been a moral support all the way long. Without you two, I could not have made this far. You have been a tremendous guidance, support and help in making me a better researcher. Cheers to your direction and production!

A special thanks to my follow-up group members, Prof. Markku Oivo and Prof. Mika Mäntylä for ensuring throughout that I was on the right track with my doctoral research and studies. I would also like to extend my thanks to Prof. Marcela Genero and Dr Gül Çalikli for dedicating time to review my dissertation. Their feedback has made an invaluable contribution towards improving the quality of my dissertation. I thank Prof. Oscar Pastor for acting as an opponent for my doctoral defence.

Heartiest thanks to Dr Ayşe Tosun. She was the first to teach me how to do research in her role of my Master’s thesis supervisor. I also got an opportunity to work with Prof. Natalia Juristo, which was a great learning experience. My deepest gratitude and affection for Dr Sira Vegas. Your evolving roles, from being my mentor, a (former) follow-up group member, a co-author and most importantly to a friend, has made priceless contributions to my journey; thank you for always being there for me. Moreover, I would like to acknowledge the Infotech Oulu Doctoral Grant and UniOGS for their financial support.

They say, good friends make the world go round. Anila, Ijaz, Iqra, Itir, Jukka, Madiha, Muzamil, Nirmaya, Ovais, Pakhtana, Pertti, Salman, Sonia and Zeeshan; thank you for adding colours to my happy and gloomy times. Usman and Hira, despite living thousands of miles away, you have always stayed connected with me for my moral and emotional support - much affection for you. Thanks to all my colleagues from M3S for making this period a worthwhile. Thanks to everyone else also who played happy notes at many points of this journey.
Mama, Midhat and Khursund, I surely wouldn’t have concluded this chapter of my life without your immense love and support. I have dedicated this thesis to my late father because all I am today is due to his love and believe in me. He has been my inspiration.

Hello to new beginning!

Oulu, Oct 2019

Iflaah Salman
List of abbreviations

ANOVA  Analysis of Variance
AP     Actual Performance
CB     Confirmation Bias
GT     Grounded Theory
NTP    No Time Pressure
PP     Perceived Performance
SMS    Systematic Mapping Study
SUT    System Under Test
SWEBOK Software Engineering Body of Knowledge
TDD    Test Driven Development
TLD    Test Last Development
TP     Time Pressure
**List of original publications**

This thesis is based on the following articles, which are referred to in the text by their Roman numerals (I–V).


Contents

Abstract                                      9
Tiivistelmä                                    9
Acknowledgements                               9
List of abbreviations                         11
List of original publications                 13
Contents                                      15
1 Introduction                                17
  1.1 Motivation and research gaps              18
  1.2 Objective and research questions         19
  1.3 Research design overview                 20
  1.4 Scope of the research                   22
  1.5 Dissertation structure                  23
2 Background and related work                 25
  2.1 Cognitive biases                         25
  2.2 Confirmation bias                        26
  2.3 Time pressure                            28
  2.4 Time pressure and confirmation bias      30
  2.5 Students vs professionals                31
  2.6 Limitations of existing research         32
3 Research process and methods                35
  3.1 Research process                         35
     3.1.1 Phase 1                             37
     3.1.2 Phase 2                             37
  3.2 Systematic mapping study                 38
  3.3 Experiment                               40
     3.3.1 Variables and metrics               40
     3.3.2 Instrumentation                     43
     3.3.3 Sampling                            45
     3.3.4 Experimental design                 46
     3.3.5 Data analysis                        49
  3.4 Grounded theory                          51
4 Original research articles

4.1 Article I: cognitive biases in software engineering: a systematic mapping study ........................................... 55
4.2 Article II: are students representatives of professionals in software engineering experiments? .......................... 58
4.3 Article III: a controlled experiment on time pressure and confirmation bias in functional software testing .................... 59
4.4 Article IV: effect of time-pressure on perceived and actual performance in functional software testing .................... 61
4.5 Article V: what leads to a confirmatory or disconfirmatory behaviour of software testers? .................. 64

5 Discussion

5.1 RQ1: what is the state of the art research on cognitive biases in software engineering (SE)? ......................... 69
5.2 RQ2: are students representative of professionals in software testing experiments? ............................... 70
5.3 RQ3: do software testers manifest confirmation bias in software testing? ............................................ 71
5.4 RQ4: how does time pressure affect confirmation bias as well as the actual and perceived performance of testers? ........ 72
5.5 RQ5: why do software testers manifest confirmation bias? .......... 74
5.6 Threats to validity .............................................................. 76
5.6.1 Construct validity ......................................................... 76
5.6.2 Internal validity ......................................................... 77
5.6.3 External validity .......................................................... 78
5.6.4 Conclusion validity ..................................................... 79

6 Conclusions

6.1 Contributions ............................................................... 81
6.2 Relevance to the industry .................................................... 83
6.3 Relevance to academia and research ...................................... 84
6.4 Future research .............................................................. 85

References ................................................................. 87

Original publications .................................................. 97
1 Introduction

Software has become a part of our daily personal and professional lives. Hence, there are social and financial impacts when software fails (Tricentis, 2017). According to Tricentis’ report on software failures, 3.7 billion of the world’s population of 7.4 billion was affected by software failures in 2017 (2017). The financial losses incurred from these failures are more than a trillion US dollars (Tricentis, 2017). A major contribution to software failures was made by software bugs (63%) aside from usability glitches and security vulnerabilities (Tricentis, 2017). Besides financial loses for businesses due to software failures, they also lose credibility (Tricentis, 2017). Based on a recent survey of 1700 executives from 32 countries across multiple industries, 42% of executives aim to detect defects before going live in production to prevent financial losses and maintain credibility in the consumer market (Capgemini, Sogeti, & Focus, 2018). These executives intend to detect defects through quality assurance and testing (Capgemini et al., 2018). Industries need a change in their software testing strategies and testers’ skills because the count of bugs released to production has continued to increase (Capgemini et al., 2018; Tricentis, 2017).

With much advancement and use of technological solutions to address the challenges of software testing, it is time that we also explore the cognitive aspects of software testing given that software testing activities are performed by software testers, and a human mind is inherent to flaws of thinking that deviate from rationality (Arnott, 1998, 2006). Arnott (2006) defines cognitive biases as “cognitions or mental behaviours that prejudice decision quality in a significant number of decisions for a significant number of people” (Arnott, 2006, p. 59). Amos Tversky and Daniel Kahneman introduced the concept of cognitive biases in the late 1960s and early 1970s (Gilovich, Griffin, & Kahneman, 2002). Since then, multiple disciplines have performed research on the phenomenon of cognitive biases and their effects, e.g. psychology, sociology, medicine, economics and law (Chapman & Elstein, 2000; Charness & Dave, 2017; Fleischmann, Amirpur, & Benlian, 2014; Gilovich et al., 2002; Glick, 2017; Hernandez & Preston, 2013). The field of software engineering (SE) is also susceptible to the negative effects of the cognitive biases of software engineers (Stacy & MacMillan, 1995).

There are several studies in SE that report the deleterious effects of cognitive biases, e.g. the effects of fixation bias in requirements (Mohanani, Ralph, & Shreeve,
availability bias in design (van Vliet & Tang, 2016), confirmation bias in testing (Calikli & Bener, 2010) and anchoring and adjustment bias in software management (Abdel-Hamid, Sengupta, & Ronan, 1993).

One of the foremost reasons for problematic human reasoning is confirmation bias (Nickerson, 1998). It is a tendency to seek for or interpret evidence that confirms one’s existing beliefs rather than look for disconfirming evidence (Arnott, 2006; Nickerson, 1998). Confirmation bias among software testers is their tendency to exercise a program with test cases that validate its correct functioning rather than to execute test cases that reveal errors (Calikli & Bener, 2014; Teasley, Leventhal, Mynatt, & Rohlman, 1994). Therefore, due to confirmation bias, higher defect rates and increased production bugs were observed by Calikli and Bener (2010) as well as Calikli, Bener, Aytac, and Bozcan (2013) in software testing. Software testing literature on cognitive biases reports multiple antecedents to confirmation bias that result in exacerbated software quality, e.g. low expertise levels of testers (Leventhal, Teasley, & Rohlman, 1994), and a lack of logical reasoning skills of software developers/testers (Calikli & Bener, 2014).

Research in applied social psychology suggests that the limitations of cognitive resources also act as an antecedent to confirmation bias, e.g. time (Ask & Granhag, 2007; Hernandez & Preston, 2013). Time pressure hampers the analytical processing of information at hand, which decreases an individual’s ability to disconfirm his/her pre-existing beliefs (Ask & Granhag, 2007). Time pressure is one of the constant environmental factors for criminal investigators, which has led to biased judgement and decision making (Ask & Granhag, 2007; Hernandez & Preston, 2013). Time pressure is also an unavoidable phenomenon of software engineering that leads to problems when engineers perceive it (Austin, 2001). In SE, it is often perceived as a factor that has negative effects, e.g. a demotivating factor for software process improvement, a source of error in a distributed development environment as well as on software quality (Baddoo & Hall, 2003; Cataldo, 2010; Mäntylä, Petersen, Lehtinen, & Lassenius, 2014; Wilson & Hall, 1998). Kuutila, Mantyla, Claes, and Eloainio (2017) identified that software testing and code quality are the areas of SE most affected by time pressure.

1.1 Motivation and research gaps

The SE literature concerning confirmation bias among software testers and its effects on software testing is limited in evidence and the applied research approaches, e.g. the qualitative approach (Mohanani, Salman, Turhan, Rodriguez, & Ralph, 2018). It
also lacks an explanation for the underlying phenomenon of confirmation bias among software testers while performing software testing.

Similar to the effects of confirmation bias, the effects of time pressure are also detrimental for software quality (Shah, Harrold, & Sinha, 2014; Wilson & Hall, 1998). The empirical evidence is also limited in this regard and, thus, is in need of further attention (Kuutila et al., 2017; Mäntylä et al., 2014). The evidence from psychology literature suggests that the combination of time pressure and confirmation bias in the software testing context may exacerbate software quality. This unexplored combination implies that there might be other antecedents to confirmation bias among software testers that are still not investigated.

We, therefore, describe the need for further investigations through two main research gaps:

RG1: To investigate the effect of time pressure on confirmation bias in software testing and how testers assess their performance as an effect of time pressure in the same context.

RG2: To explore antecedents to confirmation bias and how it occurs among software testers when they perform software testing.

1.2 Objective and research questions

According to the identified research gaps, the objective of this dissertation is to examine the manifestation of confirmation bias, why and how it occurs among software testers, the role of time pressure in confirmation bias promotion and the perceived role of time pressure.

In order to address the main objective, we derive five specific research questions and present how they relate to the research gaps, as follows:

RQ1 What is the state of the art research on cognitive biases in software engineering?

The first question helps in assessing state of the art research on cognitive biases in the SE discipline, e.g. when and how confirmation bias in software testing is examined in the literature and what gaps there are in the existing research, e.g. from the perspective of antecedents. The systematically derived findings will provide a substantial base for proceeding further with a refined approach towards addressing RG1 and RG2.
RQ2 Are students representative of professionals in software testing experiments?

It was not convenient for us to recruit professionals or to perform experiments in the industry to address the identified research gaps. Therefore, this question investigates the possibility of recruiting students in place of professionals for software testing experiments due to the generalizability issues of experiments that are run with students. The particular objective in the context of this dissertation is to ascertain in what capacity we can recruit students to answer the following research questions.

RQ3 Do software testers manifest confirmation bias in software testing?

This question is a core step in addressing RG1. It is important to find out whether testers manifest confirmation bias before examining the effect of time pressure on the promotion of confirmation bias.

RQ4 How does time pressure affect confirmation bias as well as the actual and perceived performance of testers?

This question examines time pressure’s effect on the promotion of the confirmation bias of software testers. In order to properly understand the role of time pressure, we intend to examine its effect not only on the actual performance of testers but also on how the testers self-assess their performance (perceived performance) under time pressure. The answer to this question fully addresses RG1 within the scope of this dissertation.

RQ5 Why do software testers manifest confirmation bias?

This question intends to identify antecedents to confirmation bias among software testers. The antecedents may belong to the environment, the testing process, or personal attributes as well as how the phenomenon of confirmation bias occurs due to those antecedents in software testing. The answer to this question addresses RG2 of this dissertation.

1.3 Research design overview

We employ empirical research methods to address the objectives of this research work. We use both qualitative and quantitative research approaches to answer the research questions. The research is divided in two phases, as summarised below.

Phase 1: RQ1 and RQ2 are answered in Phase 1 because their findings provide evidence to answer the rest of the research questions in Phase 2. RQ1 is answered using systematic mapping to know the state of the art research on cognitive biases. RQ2 is an
empirical analysis of software testing experiments run with students and professionals to examine the suitability of recruiting students to help answer the research questions in Phase 2.

**Phase 2:** This phase answers RQs 3, 4 and 5. The research questions not only receive input from the previous phase but also provide the necessary evidence to serve as motivation to answer each other. RQ3 and RQ4 are answered using the experiments. These RQs, respectively, examine the manifestation of confirmation bias among software testers and the role of time pressure in confirmation bias promotion and the performance of testers. The answer to RQ3 contributes to the motivation to answer RQ5. Therefore, the grounded theory research method (a qualitative approach) answers RQ5, exploring why and how confirmation bias occurs among software testers.

The five RQs are answered in five distinct original publications (Article I to Article V) in peer-reviewed forums. However, each RQ does not distinctly map onto a separate publication. The two-phased research process helped to address the objectives of this dissertation systematically yet confined to the studied contexts. Fig. 1 depicts how the research gaps link to the research questions and the original publications/articles. For example, RQ1 is answered in the original publication Article I as part of Phase 1, which substantially contributes (solid arrow) to addressing RG1 and RG2. Similarly, RQ4 is answered in Phase 2 via Article III and Article IV. Hence, it is also indicated with a solid arrow to addresses a part of RG1. RQ2’s contribution towards RG1 and RG2 is presented with a dashed arrow because it does not address the research gaps directly but, rather, contributes to helping address those by RQ3 and RQ4.

**Context**

The context of the investigation in software testing is functional software testing. The investigations were carried out in academic and industrial settings by recruiting the participants from the respective environments based on convenience. The academic participants were graduate students while the industrial participants varied in terms of years of industrial experience. The individual skills (e.g. test case designing) and tendencies (e.g. manifestation of confirmation bias) were the focus of the investigations. The cause and effect relationships were studied in controlled environments for both academia and industry, whereas the exploratory investigation was performed only in an industrial environment. The objects on which the effects were studied varied in
complexity depending upon the design of the study. The data analysed in the quantitative and qualitative studies were collected over the duration of four years from 2014 to 2017.

1.4 Scope of the research

We address confirmation bias in this research from two aspects, as a possible effect of time pressure and what other factors exist in an industrial context that lead to its manifestation. The possible effects of confirmation bias are discussed confined to the respective context of the investigations. This research work does not propose debiasing interventions to mitigate confirmation bias because proposing a debiasing intervention is in itself a separate research dimension. Instead, we propose recommendations in the form of implications for practitioners that may help to mitigate the deleterious effects of confirmation bias to the extent of the studied context.

One of the theories that define a relationship between time pressure and performance is Yerkes Dodson’s law (Kuutila et al., 2017). According to this law, there is a U-shaped relation between time pressure and performance (Kuutila et al., 2017). Kuutila et al. (2017) explained it in the context of software development that deadlines can lead to increased performance of developers, however, the performance decreases when prerequisite for deadlines become excessively high. This research work carries out its investigations considering the latter situation of the law, investigated in software testing

1“Debiasing is the process of inhibiting or removing the effects of biases” (Ralph, 2011, p. 98).
context. In other words, finding the optimal amount of time pressure to increase the performance of software testers is not in the scope of this research work. We administer time pressure psychologically to observe its effect, therefore, we study it as a perceived temporal demand. If software testers perceive time pressure, then it may affect their manifestation of confirmation bias and performance in testing. The performance is assessed as functional completeness rather than a fault detection ability as a result of test case execution. While studying confirmation bias and time pressure, we disregard individual personality traits or emotions that may also affect this phenomenon. This phenomenon is particularly examined regarding software testers either by job role or title.

1.5 Dissertation structure

The next chapter presents the relevant research work reviewed to address the objectives of each research question. Chapter 3 details the research process and the applied research methods. Chapter 4 summarises the original research articles and presents their findings. We answer and discuss the dissertation’s research questions in Chapter 5 as well as threats to the validity of our research. Chapter 6 concludes the performed research work, defines the dissertation relevance for the industry and academia, and discusses future research opportunities.


2 Background and related work

This chapter presents the related work and background from the perspective of the focus of the research questions in Chapter 1.

2.1 Cognitive biases

Human judgement for uncertain events usually relies on a modest number of simplifying heuristics instead of relying on formal and logical processing (Gilovich et al., 2002). These simplified heuristics or rules of thumb are useful because they lessen the complexity of assessing an uncertain event or quantity and offer a compelling and reasonably workable solution to an assessment problem at hand (Gilovich et al., 2002; Tversky & Kahneman, 1973). However, simplifying heuristics may result in systematic errors in decision making (cognitive biases) because the solutions are not ultimately accurate (Fleischmann et al., 2014; Gilovich et al., 2002; Tversky & Kahneman, 1973); given that "cognitive biases are cognitions or mental behaviours that prejudice decision quality in a significant number of decisions for a significant number of people" (Arnott, 2006, p. 59). Decision biases or judgement biases terminology is also used to refer to cognitive biases (Arnott, 1998).

Human reasoning is intrinsic to cognitive biases, and humans are almost incapable of recognising their own biases (Arnott, 1998, 2006; Kahneman, Lovallo, & Sibony, 2011). Kahneman et al. (2011) explained this by referring to two modes of thinking: intuitive (system-one) and reflective (system-two). In system-one, thinking about actions flows quickly and effortlessly to allow us to do our daily tasks, e.g. walking, playing a game, etc., while considering other things in parallel. System-two, on the contrary, is intentional, effortful and slow. For example, it is active when we fill a job application form. Although both modes of thinking are active all along, the reflective mode takes precedence when there is an imminent error, or we have vested interests in the task; otherwise, our thoughts are usually regulated by system-one. System-one thinking is more prone to biases because it works to make sense of our surroundings in their contexts, which ultimately suppresses other explanations. A person is unaware of system-one’s operation because it is proficient in making contextual sense. (Kahneman et al., 2011)
There are multiple studies in SE that have investigated cognitive biases (Abdel-Hamid et al., 1993; Calikli, Bener, & Arslan, 2010; Jørgensen & Sjøberg, 2000; Ralph, 2011). For example, representativeness bias is a tendency "to reduce many inferential tasks to simple similarity judgments" (Mohanani et al., 2018, p. 22). Due to representativeness, certain code features may appear more frequent even if they are infrequent but prominent for other reasons (Stacy & MacMillan, 1995). After analysing the data of a web development project and conducting an experiment, Moløkken-Østvold and Jørgensen (2005) found that practitioners in technical roles manifested optimism bias in making effort estimates compared to practitioners in non-technical roles (e.g. managers). Optimism bias is a tendency to be overly optimistic in making estimations, judgements and predictions (Mohanani et al., 2018). In the mentioned case, technical competence has acted as an antecedent to making less realistic estimates, i.e. optimism bias. Our next section discusses only confirmation bias because it is the focus of this dissertation.

### 2.2 Confirmation bias

Confirmation bias is a human tendency to look for evidence that confirms prior beliefs rather than to look for disconfirming evidence to refute those beliefs (Arnott, 2006). It is an unintentional use of evidence or shaping of facts to fit one’s hypotheses or beliefs (Nickerson, 1998). According to the taxonomy of cognitive biases proposed by Arnott (2006) for decision support systems, confirmation bias belongs to the category of confidence biases. Fleischmann et al. (2014) categorised it as a pattern recognition bias in the information systems field based on the cognitive bias categorisations of Browne and Parsons (2012); Kahneman et al. (2011); Lovallo and Sibony (2010). The classification of confirmation bias in these two examples is based on the knowledge in the respective disciplines.

Confirmation bias affects multiple areas of SE. For example, Mohan, Kumar, and Benbunan-Fich (2009) as well as de Graaf, Liang, Tang, and van Vliet (2014) found that professionals insufficiently explored the software documentation and design models and were unrealistically sure of the correctness and completeness of their search queries. This is because confirmation bias hindered their process of retrieving information from the documentation. Stacy and MacMillan (1995) have suggested to perform an empirical investigation instead of using intuition for the task at hand and to seek disconfirmatory information to mitigate confirmation bias. Similarly, system engineers may mitigate
confirmation bias in trade-off-studies (a practice that proposes a method to choose among alternatives) by evaluating multiple solutions in parallel to generating them (Smith, Son, Piattelli-Palmarini, & Terry Bahill, 2007).

Confirmation bias is also referred to as positive test bias or positive test strategy in software testing (Leventhal, Teasley, Rohlman, & Instone, 1993; Leventhal et al., 1994; Teasley et al., 1994). According to Klayman and Ha (1989), positive test bias is also referred to as confirmation bias or is one instance of confirmation bias (Leventhal et al., 1994; Nickerson, 1998). Functional software testing is similar to testing the validity of the hypothesis (Leventhal et al., 1993). It is performed by treating the system as a black-box, i.e. not testing the implemented logic but the behaviour (Leventhal et al., 1993). The objective of functional testing is to validate the actual behaviour of the system against the required specified behaviour of the system. Confirmation bias occurs if a tester validates the behaviour of the system as specified in the requirements; it is hypothesis-consistent or hypothesis confirming testing (Leventhal et al., 1993; Teasley et al., 1994).

In software testing, the main focus of the existing research is to study the antecedents to confirmation bias using controlled experiments. Leventhal et al. (1994) found that higher expertise and complete specifications may mitigate positive test bias among software testers. Their findings regarding the effect of error feedback on confirmation bias remained inconclusive (Leventhal et al., 1994). However, in another study by Teasley et al. (1994), the effect of the completeness of specifications remained inconclusive, and, though higher expertise decreased positive test bias, it did not eliminate it in functional testing.

According to Calikli and Bener (2010) and Calikli et al. (2013), confirmation bias leads to higher defect rates. Calikli and Bener (2010) also found that experience led to low confirmation bias levels but for those participants who were inactive in development/testing. Logical reasoning skills gained through education and researchers (job title) showed low confirmation bias levels (Calikli & Bener, 2010, 2014). Company culture concerning different geographical locations also affects confirmation bias levels (Calikli et al., 2010). However, education, educational background or level, company size and experience were not observed to affect the levels of confirmation bias of software developers and testers (Calikli & Bener, 2014; Calikli et al., 2010). Causevic, Shukla, Punnekkat, and Sundmark (2013), in their experiment on TDD vs TLD, found that the participants manifested confirmation bias by creating more positive test cases.
compared to negative test cases. They also found that negative test cases detect more defects than positive test cases (Causevic et al., 2013).

To measure confirmation bias, the software testing literature reports multiple methods that can be used:

- Leventhal et al. (1994) and Teasley et al. (1994) mapped positive test bias onto functional software testing as testing is referred to as hypothesis-consistent if it is performed using the data specified in the requirements specification, but, if testing is performed using the data outside of the requirements specification, then, it follows a hypothesis-inconsistent way. They measured positive test bias by assessing the type of test cases as positive or negative, as prepared by the participants. A test case is positive if it deals with the data from valid equivalence classes, otherwise, it is negative if it deals with the data from invalid equivalence classes.
- Causevic et al. (2013) also assessed the types of test cases designed by the participants. According to them, a test case was negative if it validated the program for a behaviour that was not explicit in the requirements or validated a behaviour that was implicitly forbidden.
- Calikli and Bener (2010, 2014); Calikli et al. (2010), instead of assessing the test artefacts, assessed how people think. The authors amended psychological instruments developed by Wason (1960): Wason’s Rule Discovery and the Selection Task.

These methods indicate that confirmation bias can either be measured by evaluating the test artefacts or by using the psychological instruments.

2.3 Time pressure

Time pressure is also referred to as schedule pressure and deadline pressure in the SE literature (Austin, 2001; Nan & Harter, 2009; Shah et al., 2014). Hazzan, Hazzan, Dubinsky, and Dubinsky (2007) have mentioned multiple software engineering problems that lead to creating time pressure, often towards the end of the development process, which results in ignorance of important phases, e.g. testing. These problems are: bottlenecks, which occur due to a dependency on functions in the development process; project planning, which is itself a challenge, tied with the problem of meeting the planned schedule; and correct time estimation for the development of the modules (Hazzan et al., 2007).
According to Austin (2001) and Nan and Harter (2009), schedule and budget constraints of the software development process induce time pressure. High customer demands, advanced technology and the urge of the companies to compete in the global market lead software companies to deliver in a short time for the less financial budget (Mäntylä et al., 2014; Nan & Harter, 2009). The results of the mentioned factors have constrained the schedules of the software development process (Mäntylä et al., 2014; Nan & Harter, 2009). In the present research, we did not aim to perform a systematic literature review or a mapping study on the topic of time pressure in SE because it already exists as performed by Kuutila et al. (2017). We further summarise the positive and negative effects of time pressure in SE respective to the studied contexts.

Harris, Collins, and Hevner (2009) hypothesised concerning the role of time that employees’ stress and motivation are higher when they work using agile methods that apply development iterations with short deadlines. On deriving a relationship between deadline setting policies and product quality, Austin (2001) showed that the addition of slack time does not always maintain quality. The author suggested that setting aggressive deadlines may prohibit developers from taking shortcuts, which will increase or maintain time pressure for developers (Austin, 2001). However, Nan and Harter (2009) did not find a significant U-shaped effect of time pressure on development outcomes. The authors could not observe the potential benefits of schedule pressure because they might not have been mitigated by the involvement of clients in the development (Nan & Harter, 2009).

Topi, Valacich, and Hoffer (2005) observed that time pressure did not affect the task performance in an experimental investigation of the effects of task complexity and time availability for database query tasks. However, task complexity affected performance at all levels of time availability. Similarly, Mäntylä et al. (2014) also could not observe time pressure negatively affecting the effectiveness of software testing. Moreover, the effect of knowledge was also not observed on time pressure, however, time pressure improved the efficiency of test case development and the review of requirements (Mäntylä et al., 2014). Mäntylä and Itkonen (2013) performed an experiment to observe the effects of multiple testers and time pressure on the efficiency and effectiveness of manual testing. The authors found that multiple testers and testing under time pressure delivered better defect detection results compared to the individual testers under no time pressure (Mäntylä & Itkonen, 2013).

Furthermore, time pressure was found to be a relevant source of error by Cataldo (2010) in a qualitative study on distributed development projects. Additionally, Wilson
and Hall (1998) determined software engineers’ views on quality and suspected time pressure to have adverse effects on the software development cycle. Shah et al. (2014) also found that test engineers perceive time pressure to negatively affect quality in the global software testing context. They further discovered that team configurations affect the perception of time pressure (Shah et al., 2014). Similarly, Baddoo and Hall (2003) identified time pressure as one of the demotivating factors for software practitioners in software process improvement.

2.4 Time pressure and confirmation bias

Empirical evidence from psychology literature reports that time pressure promotes confirmation bias. This section presents the relevant work from research because no study in SE has examined the effect of time pressure on confirmation bias to the best of our knowledge.

Hulland and Kleinmuntz (1994) experimentally investigated four factors, including time pressure, that encourage decision-makers to rely on the information retrieved from memory (internal summary evaluations, which are steadily assessed reactions compared to other possible alternatives) rather than information obtained from the external environment to reduce effort. The authors found that time pressure affected the search and evaluation process (Hulland & Kleinmuntz, 1994). However, increased costs of external search (one of the other three factors) caused more reliance on internal summary evaluations compared to time pressure (Hulland & Kleinmuntz, 1994).

In applied social psychology, Ask and Granhag (2007) studied the behaviour of criminal investigators in evaluating witness’ testimonies as either confirming or disconfirming and how time pressure affects the evaluation. The results showed that participants found a witness who provided inconsistent (disconfirming) evidence to their prior beliefs to be less reliable and credible (Ask & Granhag, 2007). Additionally, time pressure made the participants less effective regarding the subsequent evidence and also caused them to adhere more to their prior beliefs (Ask & Granhag, 2007). Moreover, Hernandez and Preston (2013) performed experiments to investigate whether difficulty in processing information (disfluency) decreases confirmation bias for decision-making jurors. The authors observed that disfluency could not decrease confirmation bias among participants when they were exposed to other cognitive loads that also included time pressure (Hernandez & Preston, 2013).
In a similar way to criminal investigations, confirmation bias under time pressure in software testing can also occur. For example, a juror under time pressure may decide upon a guilty verdict for the defendant after reading his objectively defined case of the alleged crime if he was previously exposed to a psychologist’s report that highlighted the negative aspects of the defendant’s personality (Hernandez & Preston, 2013). In this situation, the verdict was biased because the juror looked for the evidence in the defendant’s case that confirmed his beliefs set by the psychologist’s report. Time pressure in this situation added to the scarcity of cognitive resources that promoted the manifestation of confirmation bias. Similarly, a software tester under time pressure may also manifest confirmation bias by giving preference or only executing the test cases that confirm the specified behaviour of the program. The tester might get preconditioned to test what was specified in the requirements. In other words, due to time pressure, a tester may ignore testing the program’s unspecified behaviour that could reveal errors in it.

2.5 Students vs professionals

Controlled experimentation with students is usually a compromise concerning the external validity of the experiments (Höst, Regnell, & Wohlin, 2000; Runeson, 2003; Sjøberg et al., 2002, 2005). The generalizability of results to the industry is compromised because students in several situations are not realistic subjects, i.e. a true representation of the population for which the results of the experiments are to be generalised (Sjøberg et al., 2002). The tasks used in the experiments and the environment are also not a true representation of the industrial aspects (Harrison, 2000; Sjøberg et al., 2002). This is due to simple and short spanned tasks and the absence of the industrial infrastructure of technology in experiments run with students (Harrison, 2000; Sjøberg et al., 2002).

Researchers recruit students as subjects for experiments because students are easy to access, less expensive compared to professionals and they leverage a possibility of obtaining large samples (Fallessi et al., 2018; Harrison, 2000; Sjøberg et al., 2005). However, the survey results of Fallessi et al. (2018) showed that professionals are also a convenient sample for experiments. It is, however, challenging to execute controlled experiments in industrial environments because it is difficult to locate professionals in one place, scheduling the experimental activities and controlling other possible affecting factors (Harrison, 2000; Sjøberg et al., 2002). According to Feldt et al. (2018), it is easy to control internal validity aspects when executing experiments with students, but it does not imply that those experiments have better internal validity. Moreover, students are
recruited to validate initial hypotheses and experimental designs before experimentation in the industry (Carver, Jaccheri, Morasca, & Shull, 2003; Falessi et al., 2018; Sjoberg et al., 2005; Tichy, 2000). Falessi et al. (2018) gathered understanding concerning the pros and cons of using students and professionals and concluded that it is impossible to declare one subject type better than the other as experimental subjects. Moreover, the authors suggested that experiments should characterise participants based on their experience rather than labelling them as ‘students’ or ‘professionals’ (Falessi et al., 2018).

A few studies in SE have compared students and professionals to assess similarity or dissimilarity between the two subject groups. Porter and Votta (1998) and McMeekin, Konsky, Robey, and Cooper (2009) reported significant differences between professionals and students in the context of requirements inspection techniques and software inspection techniques, respectively. McMeekin et al. (2009) attributed the difference to the experience level of professionals because they performed better. Although, Höst et al. (2000) observed minor differences between the two subject groups when compared regarding the conception of the factors that impact project lead time. However, no difference was observed when compared to the actual importance of the factors (Höst et al., 2000). Runeson (2003), Berander (2004) and Svahnberg, Aurum, and Wohlin (2008) observed the similar performance of students and professionals for personal software process improvement, requirements prioritisation (students working in projects) and the requirements selection process, respectively.

### 2.6 Limitations of existing research

The literature presented in this section highlights certain gaps particular to the research on confirmation bias and time pressure in SE:

(i) The existing software testing literature has investigated multiple antecedents to confirmation bias but *time pressure* has lacked attention.

(ii) No study in software testing explains the phenomenon of confirmation bias, i.e. why and how it occurs among software testers. Instead, only experimental studies have investigated isolated factors that may lead to confirmation bias.

(iii) The existing literature on time pressure has assessed the effects on effectiveness, efficiency and the perceptions regarding time pressure in SE. However, the literature lacks investigations on the comparison of the actual and perceived performance of software testers as an effect of time pressure.
(iv) The results of the literature that has compared students and professionals has shown that (dis)similarity between the two groups varies depending upon the SE area. Therefore, it is important to compare the two groups in a software testing context because this area lacks in the existing investigations. Additionally, existing studies (see Section 2.5) have not applied objective assessment methods to compare the two subject groups, i.e. no tool has been used to objectively collect and assess the data.

These items map onto the two research gaps, RG1 and RG2, defined in Chapter 1 as: (i) and (iii) to RG1 and (ii) to RG2. The research question RQ2 in Chapter 1 addresses item (iv) of the above list. Because RQ2 helps address RG1, (iv) also maps onto RG1 of this dissertation.
Chapter 3 focuses on detailing the research process and the applied research methods to address the objective of this dissertation. We first introduce the research methods and then elaborate on their application with respect to the research questions.

Both qualitative and quantitative research paradigms are used with their respective empirical approaches. According to Shull, Singer, Sjøberg, and Damian (2008), the theoretical stance of the researcher and access to resources (e.g. participants) contributes to the selection of a research method. A combination of qualitative and quantitative studies aid in better understanding of a phenomenon because they complement each other (Juristo & Moreno, 2001; Shull et al., 2008). For example, quantitative studies can be used to validate hypotheses suggested by qualitative research. Similarly, qualitative research may aid in finding the causes of a set of matching ideas suggested by quantitative studies in a discipline (Juristo & Moreno, 2001). The studies performed as part of this dissertation also complement each other, which is explained in the next section.

The empirical methods used were applied by following the guidelines for conducting them in a SE context. For example, we consulted Wohlin et al. (2012) guidelines to experimentation, Shull et al. (2008) guidelines for collecting qualitative data, Kitchenham et al. (2010) guidelines for systematic reviews and Urquhart (2012) for a grounded theory approach.

3.1 Research process

The research process of this dissertation is divided into two phases, Phase 1 and Phase 2. Fig. 2 presents the research process by relating the research questions to the applied empirical method used to address them, e.g. $RQ3: EXP2$ stands for $RQ3$ addressed by Experiment2. The findings of the questions in Phase 1 provide the necessary empirical evidence for the investigations in Phase 2. The answers to the research questions in Phase 2 also provide evidence as an input to answer other research questions of the same phase. The arrows in the figure depict the particular findings of a question that serve as essential inputs to the pointed-to research question.
Fig. 2. Research process.

Key:
- TP: Time Pressue
- CB: Commitment Bias
- GT: Grounded Theory
- EX: Experiment
- ES: Empirical Study
- SMS: Systematic Mapping Study

Phase 1:
- TP on CB
- TP on GT
- TP on ES

Phase 2:
- CB vs. GT
- CB vs. ES
- CB vs. SMS

Need of exploratory studies + evidence of CB in software testing

Evidence of CB in software testing

Literature on CB in software testing

Missing anecdotes to commitment bias

Students as novice professionals

Relational literature

Fields & Psychology

From GT and other

RQ1: EX

RQ2: ES(EXP1)

RQ3: EX

RQ4: EX

RQ5: GT

WHY

WHY

WHY

WHY

WHY

WHY

WHY
3.1.1 Phase 1

RQ1 is part of Phase 1 because it sets the foundation for this dissertation. In order to know the state of the art and potential future research opportunities concerning the topic of cognitive biases in SE, we performed a systematic mapping study (SMS). The findings showed that confirmation bias was the only empirically investigated cognitive bias in software testing. The identified literature also provided evidence of confirmation bias among software testers. An examination of the literature on confirmation bias in software testing enabled us to understand how confirmation bias has been measured recently. This led to investigating confirmation bias again (RQ3), albeit from a new perspective of confirmation bias assessment compared to earlier work (Chapter 2). The other findings of RQ1 that contribute directly to the studies in Phase 2 are the antecedents to cognitive biases in SE and the effects of cognitive biases. The findings on the antecedents to confirmation bias revealed a gap concerning time pressure’s effect on confirmation bias, which led to an investigation of time pressure’s effect on confirmation bias, which is RQ4. The answers of RQ1 contributed in formulating the hypotheses to answer RQ3 regarding confirmation bias manifestation and a part of RQ4, examining the effect of time pressure on confirmation bias. The findings of the SMS also highlighted a need for conducting exploratory studies to examine how cognitive biases manifest in the software development process. This motivated RQ5 to explore what leads to confirmation bias in software testing and how.

RQ2 in Phase 1 investigates, whether students represent professionals in software testing experiments. The findings of the empirical study enabled us to recruit students as representatives of novice professionals in the experimental investigations of RQ3 and RQ4, in Phase 2. The empirical study was based on the data of an experiment (see Fig. 2 - RQ2: ES[EXP1]).

3.1.2 Phase 2

The answers to the research questions in Phase 1 provided a sufficient initiative to help answer the questions in Phase 2. The findings of the questions in this phase also provided input to the other questions of this phase. For example, the deductive findings of RQ3, based on the experiment EXP2, contributed to the need to investigate RQ5. The nature of inputs to this question motivated an inductive approach (grounded theory [GT]) to investigate what leads to confirmation bias among software testers and how. The
inputs to RQ5 are an example of quantitative research complemented with qualitative research.

To answer RQ4, the theoretical construct of time pressure’s effect on confirmation bias was supported by the relevant literature from the psychology discipline (see Chapter 2). It also helped in formulating the hypotheses of the experimental investigation - EXP2. The literature on time pressure’s effect within SE also contributed towards addressing RQ4, i.e. in formulating the hypotheses for EXP3. The findings of RQ4 concerning time pressure’s effect on confirmation bias led us to experimentally investigate (EXP3) the effect of time pressure on the actual and perceived performance of testers; Fig. 2 indicates this through the self-loop on RQ4.

3.2 Systematic mapping study

A systematic mapping study (SMS) is a type of secondary study that synthesises the research on a topic based on the available studies, called primary studies (Kitchenham et al., 2010). An SMS provides categorisation of the topic of interest and determines publication frequencies within those categories, resulting in a map of the topic (Petersen, Feldt, Mujtaba, & Mattsson, 2008). The differences between a systematic literature review and an SMS are the scope of the question, coverage on the topic, implications for practice and validity aspects of the study itself (Kitchenham et al., 2010; Petersen et al., 2008).

In order to answer RQ1, we performed an SMS because of the breadth of our research question - to know the state of the art research on cognitive biases in SE. We followed the guidelines by Petersen, Vakkalanka, and Kuzniarz (2015) and Kitchenham et al. (2010) concerning the performance of mapping studies. The process starts with the definition of research questions for the mapping study. Our imminent questions were to know: which cognitive biases have been investigated, the antecedents to cognitive biases, the effects of cognitive biases, the investigated debiasing techniques and which knowledge areas of SE have been targeted by research.

To search for articles on the topic, we defined a search string: software AND "cognitive bias". We ran the search-string for the literature search using five online databases with filters applied to the discipline categories. The produced search results were stored in RefWorks\(^2\). We then applied automatic duplicate removal in RefWorks and manual duplicate removal of the stored results. The next step is the screening

\(^2\)http://www.refworks.com
of relevant articles that required a definition of inclusion/exclusion (I/E) criteria. We finalised the criteria in multiple steps and calculated inter-rater agreement at each step for establishing a common understanding of the concepts among the involved authors.

In order to ensure sampling adequacy, we first performed iterative backward snowballing with the identified primary studies, which resulted in the inclusion (I/E criteria applied) of additional primary studies. Secondly, we reviewed publication records of those authors who appeared in five or more articles in the primary studies set. The author’s analysis also resulted in the addition of articles. However, backward snowballing with these additions did not lead to any new primary studies. Furthermore, we executed an unfiltered search on Google Scholar with the same search string. The first 150 results returned were manually examined, and no new articles were found. As the last step to ensure sampling adequacy, we decided to search our chosen five databases with specific bias names. We derived a list of these biases by deriving the difference between the set of the biases we found in our primary studies and the cognitive bias list formed by Fleischmann et al. (2014) for the information system field. The executed search string with the difference set resulted in the inclusion of one more study.

The next phase of the process is to perform a quality assessment of the primary studies identified until the last step. The devised quality assessment criteria were also refined in two stages to establish common grounds of the criteria among the involved authors. We assessed the quality of the primary studies based on their reporting practices (on a binary scale [yes or no]) rather than evaluating their methodological rigour. The quality assessment phase resulted in the rejection of one study because the article did not satisfy 50% of the quality criteria.

The last stage of the process is to extract the data to answer the mapping questions. We used NVivo\(^3\), a qualitative data coding and analysis tool, to extract the data from the final set of primary studies. We extracted the studied cognitive biases, antecedent to the biases, effects of the biases, debiasing techniques, utilised research methods, the publication year of the primary studies, the publication avenue of the primary studies (source), authors’ names, and the knowledge areas of SE based on software engineering body of knowledge (SWEBOK) version 3.

\(^3http://www.qsrinternational.com\)
3.3 Experiment

Experiments are a way to control the situation for rigorous and controlled investigations by manipulating the behaviour of identified factors (Wohlin et al., 2012). For this reason, they are often performed in a laboratory environment (controlled experiments), in contrast to the real world, where it is difficult to acquire strict control of the situation (Juristo & Moreno, 2001; Wohlin et al., 2012). Experiments are classified as a quantitative research approach because they gather numerical data from the studied variables. Statistical methods are applied to analyse the collected data from experiments to achieve formal results (Juristo & Moreno, 2001; Wohlin et al., 2012).

In a controlled experiment, a hypothesis (formed from the theory) is tested by manipulating the independent variables to observe their effects on the dependent variables (Shull et al., 2008). Juristo and Moreno (2001) mentioned this in simple terms of comparing theoretical speculations with reality. Controlled experiments help in determining a cause-effect relationship between the studied variables, i.e. whether the independent variable (cause) varies (effect) the observed dependent variables (Shull et al., 2008; Wohlin et al., 2012). Experiments offer execution control, measurement control, high ease of replication, albeit, with high execution costs (Wohlin et al., 2012).

Conducting an experiment proceeds through a defined process. The stages of the process are definition, planning, operation, analysis and interpretation, and reporting (Wohlin et al., 2000). RQs 3 and 4 were answered by executing experiments, whereas RQ2 was answered by quantitatively analysing the data collected from another experimental study. Therefore, for RQ2, we report the details of the empirical study and the relevant details of the experiment that were used or supported the empirical study. We followed the experimentation process defined by Wohlin et al. (2000). The following sections report the essential entities that belong to the planning and analysis phases of the experiments and how we realised them to answer the respective RQs.

3.3.1 Variables and metrics

In experimentation terminology, the independent variable is a factor, and the value(s) of the factor (levels) is(are) known as treatment(s) (Juristo & Moreno, 2001; Shull et al., 2008; Wohlin et al., 2012). The dependent variable is also known as a response variable (Juristo & Moreno, 2001). The effect of the treatment(s) is measured on the dependent variable(s) (Wohlin et al., 2012). The group on which the treatment is applied is referred
to as the experimental group, whereas the group on which the treatment is not applied is referred to as the control group. The measure of the response variable(s) has to be quantified to enable the application of statistical techniques for analysing the effects of the independent variable (Juristo & Moreno, 2001).

Relating to RQ2, the investigation of professionals representation by students is an empirical study based on the data of EXP1 (Fig. 2). We report the variables and metrics of our empirical study because of the focus of the research question. The independent variable of the empirical study was the type of Subjects with two levels: students and professionals. The dependent variable was the code quality. We measured the internal code quality for the tasks implemented by students and professionals. The internal code quality is defined as the design quality of a software system (Shull et al., 2010; Turhan, Layman, Madeline, Erdogmus, & Shull, 2010). Twenty internal code quality attributes were used as metrics for measuring the code quality of the tasks - per that extracted by Prest (introduced in the next section). The code quality for the tasks was measured on a method level for each participant (of both groups) due to the nature of the tasks’ implementation. The method-level measurement was transformed into a participant-level measure, i.e. each participant finally had five values for every metric. This transformation was important because every participant implemented a varying number of methods for the assigned tasks.

The investigation of confirmation bias among software testers (RQ3) and the effect of time pressure on confirmation bias promotion (a part of RQ4) was investigated with the same theoretical construct, i.e. using a single experiment (EXP2 in Fig. 2) for both purposes. Therefore, we report the details of the variables and the metrics of RQ3 and the relevant part of RQ4 together.

The independent variable or factor was time pressure with two treatments: time pressure (TP) and no time pressure (NTP). The values/duration for the treatments were determined based on a pilot run, which yielded 45 minutes, the average time taken by the pilot-run participants to complete their task. Based on this, we decided to allocate 30 min for the TP (treatment group) and 60 min for the NTP (controlled) group to operationalise time pressure. We psychologically administered time pressure by announcing the timings of the task differently to both groups. This operationalisation of time pressure is in line with Ask and Granhag (2007) and Hernandez and Preston (2013).

The three response variables of the study were: the number of Consistent test cases (c), the number of Inconsistent test cases (ic) and the Temporal Demand. In the context of functional software testing, a consistent test case refers to a test case that validates the
behaviour of the software program for the specified inputs or for the specified invalid inputs. An inconsistent test case refers to a test case that (1) tests the program for not explicitly stated behaviour and (2) tests the outside-of-the-box aspects relevant to the context/domain of the program. The outside-of-the-box aspect considers the completeness of the requirements’ specification.

Our definition of (in)consistent is in contrast to Leventhal et al. (1994); see Section 2.2. We do not classify a test case as inconsistent if it tests the data from an invalid equivalence class, especially when it is specified in the requirements. Instead, we refer to it as consistent because the tester is confirming what is specified in the requirements. In comparison to Causevic et al. (2013), the classification of test cases per their definition of positive and negative might yield the same results as our definition of consistent and inconsistent. However, outside-of-the-box thinking is an additional aspect of our inconsistent definition. Moreover, our assessment of confirmation bias (explained further) is based on the assessment of a test artefact, which is in contrast to Calikli and Bener (2010).

A proxy measure for assessing confirmation bias was derived based on the $c$ and $ic$, which is

$$z = c/C - ic/IC.$$ 

The value of $z$ is a rate of change in relative terms. It is the difference between consistent test case coverage and inconsistent test case coverage (Salman, Turhan, & Vegas, 2018). In the above equation, $C$ is the total number of consistent test cases and $IC$ is the total number of inconsistent test cases. A baseline test suite was developed based on the requirements specification used as an experimental object. This (heuristic) baseline provided the total count of consistent test cases ($C$) and inconsistent test cases ($IC$). The 0 value of $z$ is an indication of complete test coverage in terms of consistent and inconsistent test cases with no confirmation bias manifestation.

For the third response variable of this study, temporal demand, we applied the same definition as the temporal demand attribute of NASA-TLX (introduced later in Section 3.3.2). It is stated as a degree of time pressure experienced by the task performer due to the pace at which events occur. NASA-TLX measures temporal demand through an 100 point scale from low (0) to high (100). We used the same scale to measure the third response variable.

The effect of time pressure on actual and perceived performance (a part of RQ4) was investigated with another experiment, i.e. EXP3 in Fig. 2. The factor
was time pressure with two treatments: TP and NTP. The treatments had the same values/duration as for EXP2. The response variables were: perceived performance, actual performance and temporal demand. We used the definition of the performance attribute of NASA-TLX to measure the perceived performances of the participants. NASA-TLX assesses performance as a measure of how successful and satisfied one is with his performance in accomplishing the objectives of the task. Similar to other load attributes of NASA-TLX, it is also measured on a 100 point scale from 0 (failure) to 100 (perfect). Actual performance is measured as a percentage of coverage given to the requirements specification, i.e.

$$\text{Actual Performance} = \frac{\text{total test cases designed by the participant}}{\text{total test cases in baseline test suite}} \times 100$$

(Salman & Turhan, 2018). The third response variable, temporal demand, was measured the same way as measured in the EXP2 study.

### 3.3.2 Instrumentation

A monitored execution of the experiment without affecting the control is possible through instrumentation (Wohlin et al., 2012). Multiple types of instrumentation are involved in the experiments depending upon the study’s objectives (Wohlin et al., 2012). For example, appropriate objects are needed concerning which the experiment is run, guidelines for executing the experimental protocol and instruments (measurement forms and tools) for data collection that can be assessed per the experimentation objectives (Juristo & Moreno, 2001; Wohlin et al., 2012). Below, we first elaborate on the entire instrumentation, which is then related with the research questions and their respective experiments.

**Experimental objects**

Multiple requirements specifications of varying complexity of different programs were used as experimental objects in the studies. These included: bowling scorekeeper (BS), mars rover (MR), sudoku (S) and musicFone/musicPhone (MP). These objects can be divided into two categories based on their complexity: toy objects and realistic objects. The first three objects are greenfield toy objects with similar complexity. The specifications are written in an explicit user-story mode, and no code-base existed for these programs. MusicFone is a realistic brownfield object because it had an existing
code-base. The document of this program contained instructions on how to run the existing code and specifications for adding new functionality and modifying the existing functionality from development point of view. In order to use this MusicFone document as an actual requirements specification document, we modified it to make it also usable for designing test cases.

*Prest*

Prest is an open source tool, developed by Kocagneli, Tosun, Bener, Turhan, and Çağlayan (2009). The tool extracts source code metrics and generates call graphs in five different programming languages.

*Pre-questionnaires*

Pre-questionnaires were developed to collect the background information of the participants. Depending upon the objectives, we collected information on the participants’ experience related to programming and testing in Java (development) and testing in general within academia and industry.

*NASA task load index*

The NASA Task Load Index (NASA-TLX) was developed by the Human Performance Group at the NASA Ames Research Center. It is a subjective workload assessment procedure based on a multi-dimensional rating of six workload factors: physical demands, mental demands, temporal demands, performance, frustration and effort (Research Group & Ames Research Center, 1987). The procedure determines the workload of the specific task as experienced by the task-performer (Research Group & Ames Research Center, 1987).

*Test case design template*

A test case designing template was developed to guide the participants for designing test cases. For example, what other aspects were required along with the description of a test case and the level of detail required in describing a test case.
The **EXP1, of the empirical study of RQ2**, used all the experimental objects for the empirical study. The task of the participants was to develop the programs using the experimental objects (requirements specifications) according to the treatment of EXP1. The treatments of EXP1 were the development practices: test last development (TLD) and test-first development (TDD). If the participants were assigned a toy object, they had to develop the program from scratch. In the case of the realistic object (MusicPhone), they had to modify the existing code by using its development-oriented specifications.

We then used Prest to extract the data, i.e. code quality metrics from the developed programs. This data extraction was part of our empirical study to answer RQ2.

**The EXP2 to answer RQ3 and a part of RQ4** used only one experimental object, MusicFone. The task of the participants was to design functional test cases of MusicFone. Therefore, we used the modified requirements specification document in this experimental study along with the provision of the screenshot of the application. The screenshot was provided to help understand the requirements. We used NASA-TLX as a post-questionnaire for assessing the temporal demand experienced by the participants. Since the objective of RQ4 was to examine the effect of time pressure, the subjective assessment of temporal demand served as a sanity check for the operationalisation of the time pressure construct. The test case design template was used by the participants to design functional test cases that later provided the required data. A detailed script guiding through the steps of the experiment along with the allocated timings was developed to ensure a valid experimental execution.

The instrumentation for **EXP3 to answer the effect of time pressure on the performance part of RQ4** was the same as EXP2. We used the same experimental object, pre-questionnaire, script, NASA-TLX, and test case design template. The difference existed in the data that we extracted from the designed test cases and NASA-TLX. The data extraction was according to the metrics, actual performance and perceived performance.

### 3.3.3 Sampling

The selection of participants or subjects from a population is known as sampling (Wohlin et al., 2012). It is important to choose a representative sample of the population for an experiment, thereby, the results can be generalised to the targeted population (Wohlin et al., 2012). We drew samples from the industry (professionals) and academia (students) to
answer our research questions. The selected samples, along with the applied techniques for sampling, are described briefly in the following sections.

For the EXP1 for RQ2, the students were recruited from the Universidad Politécnica de Madrid (UPM), Spain, based on a convenience sampling technique. They were enrolled in a graduate-level, five-day seminar course. There were seventeen students of different nationalities forming a culturally heterogeneous group. The student participants were trained on the TDD and TLD development practices before their formal participation in the experimental sessions. The second was from the industry based on convenience sampling. The twenty-four professionals belonged to three different sites of the same company: Helsinki (Finland), Oulu (Finland) and Kuala Lumpur (Malaysia). The company operates in the domain of security services and products. The professionals were also trained for the TDD development technique before the experimental sessions. They were not trained for TLD because they were already applying TLD for the development. We ensured confidentiality and anonymity of the participant groups (students, professionals) by not revealing their identification particulars in the reporting of the study.

We recruited students for the RQ3 and RQ4 experimental investigations (EXP2, EXP3). The student sample was drawn using convenience sampling from the Software and Quality Testing course offered for an international graduate degree programme at the University of Oulu. In total, eighty-eight students participated in the two experimental executions (year 2015: 43 and year 2016: 45). The students were taught and trained on functional testing and how to design functional test cases, per the curriculum of the course, before the experimental sessions. The participation in the experiments was voluntarily, therefore, students of the course provided signed consent forms for their work to be considered as experimental data. All the enrolled students performed the experimental activity, but only those who consented to have their data used for the research were considered as participants. We encouraged participation by offering bonus marks as an incentive, which was announced in the introductory lecture of the course. The discussion of voluntary participation vs. incentive is part of Chapter 5. Confidentiality was ensured by anonymously using their data.

### 3.3.4 Experimental design

An experiment is designed based on the aim of the experiment, the identified factor(s) (independent variable), levels of the factors and the dependent variable(s) (Juristo
Table 1. Experimental design - EXP1.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industry</td>
<td>MR</td>
<td>BSK</td>
<td>MP</td>
<td></td>
</tr>
</tbody>
</table>

& Moreno, 2001). The practicalities of executing the experiment also influence the selection of an experimental design. The designs of the experiments are described below.

**Design for the first experiment - EXP1**

The design of the experiment used to answer **RQ2** was one factor with two levels within-subjects experimental design. The same design was followed for executing the experiment in the academic and industrial settings. The treatments (development practices: TLD [A], TDD [B]) were also applied in the same sequence in both the settings. However, one level of the factor, TDD, was applied twice in the industrial setting following the order ABB, whereas it was applied thrice in the academic setting with an ABBB sequence. Hence, experimental sessions spanned over three days in the industry and four days in academia with the same set of participants in each session. In the academic setting, the participants worked on the randomly assigned experimental objects in each session. On the contrary, one object was chosen for each session in the industrial setting. Table 1 presents the experimental design for investigating RQ2.

Based on the design, we compared the programming tasks of students and professionals in two ways: a comparison in terms of treatment, e.g. TLD-students vs. TLD-professionals and a comparison in terms of treatment-object combination, e.g. TLD-MR-students vs. TLD-MR-professionals. The students and professionals were compared for internal code quality to investigate whether students are representatives of professionals.
**Design for the second experiment - EXP2**

The experimental design for investigating the manifestation of confirmation bias (RQ3) and the effect of time pressure on confirmation bias (RQ4) was one factor with two levels between-subjects experimental design. The factor was time pressure with two levels: time pressure (TP) and no time pressure (NTP). We used one experimental object (MusicFone), as mentioned before. The simplicity of the design enabled us to investigate the phenomenon without having any potential effects of other interventions, e.g. the type of the experimental object leading to task-treatment interaction. We randomly assigned 43 participants to the two experimental sessions, TP and NTP, which were run in parallel, per the design.

This experimental design was also proficiently adaptable with the logistics of the course because the experiment was run in one of the lectures. The data (designed consistent and inconsistent test cases) of the participants of the TP group was compared with the data of the participants of the NTP group to observe the effect of time pressure on confirmation bias. The participants of the two groups were also assessed for their manifestation of confirmation bias based on the same data.

**Design for the third experiment - EXP3**

To investigate the effect of time pressure on the actual and perceived performance (RQ4), we internally replicated the EXP2 (executed in 2015) with the 2016 session of the same course. This introduced the experimental session factor, with two levels, in the design of EXP2. It made the design of EXP3 a between-groups factorial design (independent factorial design). Table 2 shows the design: E is an experimental session, TP is time pressure and NTP is no time pressure. The experiment followed the same protocol from the assignment of participants to the experimental execution with the same instrumentation.

We compared four groups with the distinctive set of participants in each, as shown in Table 2, firstly, to observe the effect of time pressure on actual performance and, secondly, on perceived performance. Actual performance and perceived performance data were measured as explained in Section 3.3.1 using the designed test cases and

---

4Republished with permission of Association for Computing Machinery, from Effect of Time-pressure on Perceived and Actual Performance in Functional Software Testing, Salman & Turhan, 2018; permission conveyed through Copyright Clearance Center, Inc.
Table 2. Experimental design - EXP3 (Salman & Turhan, 2018).

<table>
<thead>
<tr>
<th>E</th>
<th>TP</th>
<th>NTP</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>Group 1</td>
<td>Group 2</td>
<td>MusicFone</td>
</tr>
<tr>
<td>E2</td>
<td>Group 3</td>
<td>Group 4</td>
<td></td>
</tr>
</tbody>
</table>

NASA-TLX. Our main factor of interest was time pressure, therefore, we did not observe the potential effect of the interaction of the experimental session with time pressure. We also investigated the relation between the actual performance and perceived performance of the participants.

3.3.5 Data analysis

In this section we present the data analysis techniques that we utilised to analyse the data collected from the experimental studies. We used statistical significance tests to interpret the results of our experimental research questions. Before executing the statistical significance tests, we also gathered descriptive statistics. We used R-Studio\(^5\) for analysing the data.

Descriptive statistics

Descriptive statistics help understand the nature of the quantitative data before conducting statistical significance tests (Field, Miles, & Field, 2012; Wohlin et al., 2012). It involves describing and graphically plotting the data for observing its distribution, i.e. how concentrated or spread out the data is on certain measurement scales (Wohlin et al., 2012). We gathered the minimum, maximum, mean, median, mode and standard deviation of our data sets. Boxplots (box-whisker diagrams) were used to graphically plot the dispersion and skewedness of the data. All descriptive statistics were reported for RQ2, RQ3 and RQ4. The graphs were plotted for RQ3 and RQ4.

Correlations

Correlations were computed to assess whether actual performance correlates with perceived performance for RQ4. We computed three types of correlation coefficients.

\(^5\)An IDE for R scripting language for statistical analysis and graphical plotting. [https://www.rstudio.com](https://www.rstudio.com)
Kendall’s tau is a non-parametric correlation coefficient and is computed when the data set is small and has many tied ranks. We also computed partial correlations (first partial and second partial) to ascertain the true correlation between the two variables by controlling for the possible effect of the variables, time pressure and experimental session. (Field et al., 2012)

Statistical significance tests

In order to test the null hypotheses that we formulated based on the theoretical constructs, we performed statistical significance tests. A null hypothesis is rejected with a given significance level - $\alpha$. An appropriate statistical significance test is chosen based on the experimental design and satisfying the assumptions of the test concerning the collected-data. Parametric statistical tests are run when the required assumptions are met, otherwise, the non-parametric counterpart of the respective test is run. We used the following significance tests of the "t-test and F-Test family:

1. Two-sample Kolmogorov-Smirnov (for the goodness of fit) test, a non-parametric test that "evaluates the differences in the general shape of the distributions between two samples (differences in skewness, etc.)" (Salman, Misirli, & Juristo, 2015, p.671).
2. Two-sample t-test, a parametric test that compares the means of the two samples.
3. Two sample Mann-Whitney U test, a non-parametric variant of the t-test.
4. Wilcoxon signed-rank test, a non-parametric variant of the one-sample t-test.
5. Two-sample Hotelling’s $T^2$, a multivariate variant of the t-test.
6. Two-way independent ANOVA (Analysis of Variance), a parametric test of the F-Test family executed for the independent (between groups) factorial design of the experiment.

In order to test the univariate and multivariate normality assumptions of the significance tests, we used the Shapiro-Wilk test. Bartlett’s test was used to test the common variance-covariance matrix assumption of Hotelling’s $T^2$. Levene’s test was used for examining the homogeneity of the variance assumption of the two-way ANOVA. We also applied the Bonferroni correction whenever it was needed.

In order to answer **RQ2**, we ran a Kolmogorov-Smirnov test for the goodness of fit to compare the code quality of students with the code quality of professionals. The chosen $\alpha$ was 0.05 to refute the null hypotheses. We answered **RQ3** by running the Wilcoxon signed-rank test, Hotelling’s $T^2$, the t-test and the Mann-Whitney U tests with
\( \alpha = 0.05 \) were run for observing the effect of time pressure on confirmation bias. The effect of time pressure on actual and perceived performance, RQ4, was tested by running ANOVA and the Mann-Whitney U tests. \( \alpha \) was 0.01 for the statistical significance tests. We executed a two-way ANOVA with blocking of the factor - experimental sessions because we were only interested in observing the effect of time pressure on actual and perceived performance.

**Effect size**

Effect size enables a quantitative description of the size of an effect rather than only establishing its statistical significance (Fritz, Morris, & Richler, 2012). It helps to understand the magnitude of the observation (the effect), hence, defining the practical or theoretical importance of the observation (Fritz et al., 2012). Table 3 shows the effect sizes reported for the executed statistical significance tests. The magnitudes in the table are written in the order of small, medium and large.

**Table 3. Effect size.**

<table>
<thead>
<tr>
<th>Effect Size Measure Type</th>
<th>Statistical Test</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohen’s ( d )</td>
<td>Kolmogorov-Smirnov; t-test</td>
<td>0.2, 0.5, 0.8</td>
</tr>
<tr>
<td>Correlation Coefficient ( r )</td>
<td>Wilcoxon signed-rank; Mann-Whitney U</td>
<td>0.10, 0.30, 0.50</td>
</tr>
<tr>
<td>Mahalanobis Distance</td>
<td>Hotelling’s ( T^2 )</td>
<td>0.25, 0.5, &gt; 1</td>
</tr>
<tr>
<td>Omega-squared ( \omega^2 )</td>
<td>Two-way ANOVA</td>
<td>0.01, 0.06, 0.14</td>
</tr>
</tbody>
</table>

### 3.4 Grounded theory

The grounded theory (GT) method is used to generate a theory that is grounded in the data (Urquhart, 2012). The theory is generated inductively and systematically from the data (Urquhart, 2012). GT is useful to understand that how a certain phenomenon occurs or is interacting with the individuals (Stol, Ralph, & Fitzgerald, 2016; Urquhart, 2012). A few key elements of GT include (Stol et al., 2016; Urquhart, 2012):

- The researchers should limit exposure to existing literature so that they are not biased by the existing established concepts or theories.
– Data analysis begins systematically as soon as it is available.
– The concepts evolve through a constant comparison method (CCM). CCM refers to constantly comparing the labelled data within one concept with the labelled data of the same or other concepts (Boeije, 2002; Urquhart, 2012).

The generated theory can be reported as a set of propositions or in a narrative framework (Urquhart, 2012).

In order to answer RQ5, we applied GT because we aimed to understand why and how software testers manifest confirmation bias. In order to explain the phenomenon of the manifestation of confirmation bias, we identified antecedents to confirmation bias in software testing. The antecedent can belong to the testing process, e.g. reviews, exist in the environment or be a personal attribute, e.g. experience. We applied the Glasserian version of GT.

We used semi-structured interviews to collect our data. We interviewed twelve professionals who work in an ICT company. The company was chosen based on our convenience. The twelve software testers were part of different projects or domains at the company and varied in their software testing experience. The sample consisted of both manual and automation test engineers, though most were manual test engineers. On average, the testing experience of our participants was approximately 6 years.

The interviews were conducted online in October 2017 via Skype. It took 65 min on average to interview each participant. We collected 13 hr (approx.) of audio and 127 pages of verbatim transcribed data.

We followed the guidelines of Urquhart (2012) and Boeije (2002) to systematically perform the data analysis. The applied data coding techniques were: open coding, selective coding and theoretical coding. These coding techniques were carried following the CCM. We also performed memoing that helped us in theoretical coding. The result of these techniques led to the identification of antecedents, classification of antecedents into categories and the relationships between the categories that evolved from the antecedents. We used NVivo to analyse our data. The unit of analysis, per our objectives, was a software tester. The results of the data analysis based on the antecedents helped us to understand why and how confirmation bias occurs among software testers.

In order to enhance the validity of our data analysis, we performed multiple steps to ensure intercoder reliability and intercoder agreement. As a first step, we performed a pilot coding of a randomly chosen interview. Four researchers took part in the pilot coding. The results of the pilot led to a refined definition of an antecedent,
the classifications of antecedents and decisions concerning which data to discard. Additionally, the process of pilot coding provided sufficient input for a single coder (researcher) to continue coding the rest of the data independently. The next phase was to code all the interviews with only a single researcher (the author of this dissertation). This phase was performed in three iterations. After each iteration, we held a joint discussion session in which confusions regarding coding and analysis were resolved. These steps ensured that coding produced by the single researcher was reliable.

3.5 Materials

The materials used in the empirical studies can be accessed from the following URLs:

- Task descriptions used for students and professionals construct (ES[EXP1]):
  http://doi.org/10.5281/zenodo.3341704
- Package for time pressure construct investigations (EXP2, EXP3):
  https://doi.org/10.5281/zenodo.1193955
- Interview script to collect qualitative data (GT):
  http://doi.org/10.5281/zenodo.3376920
4 Original research articles

This chapter summarises the original articles that are part of this dissertation to address its objectives. The articles (I - IV) are published in Julkaisufoorumi’s approved peer-reviewed forums, conferences and journals. Julkaisufoorumi (JUFO) is a ranking of academic publication forums by the Finnish scientific community for quality assessment of academic research. The rankings are: 1 - basic level, 2 - leading level and 3 - highest level. The JUFO ranking of the publication forums of the original research articles are: JUFO Level 3 for IEEE Transactions on Software Engineering and Empirical Software Engineering journals; JUFO Level 2 for the International Conference on Software Engineering and JUFO Level 1 for the International Conference on the Software and Systems Process.

The summary of each article presents its objective(s), the research method, the findings, contributions to the dissertation and the author’s contribution to the respective study (article). Chapter 3 has already detailed the applied research methods, data collection and analyses techniques, hence, these are not part of the summaries in this chapter. Table 4 presents an overview of the included original research articles. Due to the collection of qualitative and quantitative data, we also mention the setting of the study, i.e. academia or industry, in the third column of the table.

4.1 Article I: cognitive biases in software engineering: a systematic mapping study

The purpose of this study was to systematically map the state-of-the-art research on cognitive biases in SE. We aimed to find: the cognitive biases explored in SE; the reported effects, antecedents and debiasing strategies to cognitive biases; the prolific authors in the research area; the publication trend; the most used research methods and the most investigated knowledge areas in SE.

Our search string retrieved 826 articles from the chosen databases (IEEE Xplore, Scopus, Web of Science, the ACM Digital Library and Science Direct) between 1990 and 2016. This retrieved set went through de-duplication and inclusion/exclusion criteria, leading to a set of 40 primary studies. In order to further ensure the sampling
Table 4. Summary of the original research articles - not applicable (NA), academia (A), industry (I).

<table>
<thead>
<tr>
<th>Article</th>
<th>Purpose</th>
<th>Setting</th>
<th>Findings</th>
<th>RQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>To review, summarise and synthesise the state of the art research on cognitive biases in SE.</td>
<td>NA</td>
<td>We found 65 articles published between 1990 and 2016 that investigated 37 cognitive biases. We summarised the studies concerning antecedents, effects and debiasing techniques of cognitive biases. The studies were disconnected and debiasing techniques lacked attention.</td>
<td>RQ 1</td>
</tr>
<tr>
<td>AII</td>
<td>To compare students with professionals to understand how representative of professionals they are in SE experiments in the context of test driven development (TDD).</td>
<td>A, I</td>
<td>Professionals performed similarly to students when they applied the development practice in which they were not experienced (TDD). However, their performance differed from students in the case of TLD.</td>
<td>RQ 2</td>
</tr>
<tr>
<td>AIII</td>
<td>To investigate whether software testers manifest confirmation bias and the role of time pressure in its promotion.</td>
<td>A</td>
<td>Testers manifested confirmation bias but time pressure was not observed to promote confirmation bias.</td>
<td>RQ 3, 4</td>
</tr>
<tr>
<td>AVI</td>
<td>To investigate the effect of time pressure on the perceived and actual performance of testers and to analyse the correlation between the two performances.</td>
<td>A</td>
<td>Time pressure affected actual performance but the effect on perceived performance could not be observed. Perceived and actual performance negatively correlated with each other.</td>
<td>RQ 4</td>
</tr>
<tr>
<td>AV</td>
<td>To explore the antecedents to (dis)confirmatory behaviour and why and how the behaviour occurs among software testers.</td>
<td>I</td>
<td>We identified 20 antecedents to (dis)confirmatory behaviour. They are classified in nine categories. The categories time and experience influenced multiple other categories.</td>
<td>RQ 5</td>
</tr>
</tbody>
</table>

For adequacy, we performed backward snowballing on 40 primary studies. This resulted in the inclusion of 16 more studies. Another step, the author’s analysis (reviewing the publication record of the prolific authors), added nine more studies to the 56 primary studies. Finally, a search based on specific cognitive biases (the difference set of cognitive biases - see Section 3.2) added one new study to our set, resulting in 66 primary studies. Finally, the quality assessment step removed one study, which formed the final count of our primary studies, i.e. 65.

The findings from the primary studies are:

– Thirty-seven cognitive biases have been investigated in SE. The leading investigated biases are anchoring bias, confirmation bias and overconfidence bias. Most
primary studies belong to the category of interest biases when classified according to Fleischmann et al. (2014)’s taxonomy.

– The primary studies investigated the antecedents to 11 cognitive biases with most of the antecedents identified for anchoring and adjustment bias. The effects of 11 cognitive biases were reported (the set is not the same as that of the antecedents); most of the effects are reported for confirmation bias. Debiasing techniques are reported for only five cognitive biases with the most investigated for anchoring and adjustment bias.

– The most used research methods were experiments (30 primary studies) and case study (six primary studies).

– The first article was published in 1990, and 2010 was the peak year when 10 articles were published. Fifty-eight percent of the primary studies were published in journals or academic magazines. Magne Jørgensen, Ayşê Bener and Gül Çalikli have been the leading authors on the subject area.

– Software engineering management and software construction (SWEBO knowledge areas) have been the most investigated areas in SE from the cognitive biases research perspective.

– The research on cognitive biases in SE has been disconnected, i.e. most of the biases have been studied in isolation and posses inconsistent terminology issues (using different terminology to refer to the same bias).

Based on the above findings, the SMS suggests a need for more exploratory and multimethodological research to understand how cognitive biases are manifested in practice. Additionally, SE researchers are required to pay attention to the investigation of those cognitive biases that are investigated in IS but not in SE. The finding of the antecedents to cognitive biases revealed time pressure as a missing antecedent from the perspective of such an investigation. The SMS also yielded confirmation bias as the only investigated cognitive bias in software testing. Moreover, the findings showed that only controlled experiments were used to investigate confirmation bias in software testing.

Article I contributes to the dissertation by answering RQ1. This study highlighted a need for Article III by providing the list of investigated antecedents to cognitive biases and the literature on confirmation bias in software testing. The identified literature provided evidence of confirmation bias among software testers and how confirmation bias has been assessed. Furthermore, the findings motivated further exploratory research to understand why and how cognitive biases are manifested. In the context of the
dissertation, it motivated us to understand the phenomenon of confirmation bias using an exploratory approach, which leads to Article V. The author of this dissertation contributed equally to the first author of this article in the planning, executing each phase of the research method and reporting of the study.

4.2 Article II: are students representatives of professionals in software engineering experiments?

The purpose of this empirical study was to understand the propensity to recruit students as representatives of professionals for experimental participants in SE experiments. This study is based on the data from two experiments that were conducted in academic and industrial settings with students and professionals, respectively. The objective of the experiments was to observe the effects of TDD and iterative test last development (TLD) on quality and productivity. The development practices, TLD and TDD, differ in their approach to unit testing.

We extracted the data through Prest (see Section 3.3.2) from the data of the development practice experiments, run with 17 students and 24 professionals. The objective was to compare the two types of participants in terms of the internal code quality of their implemented tasks in TDD and TLD. Four different experimental objects were used for implementation in the TDD and TLD approaches by the group of students, whereas three objects were used by the professionals. We analysed the data of 14 students and 21 professionals because we dropped the data of those who did not attend all the experimental sessions.

The results were yielded from two types of comparisons due to the experimental design - explained in Section 3.3.4. The results of the first comparison (experiment treatments) revealed that the internal code quality of students differed from professionals for a TLD approach. For a TDD approach, the code quality of the two groups was similar. This study suggests that professionals differed from students when both were compared concerning the approach in which professionals were more experienced (TLD). For example, the cyclomatic complexity of the methods developed by professionals was lower than that developed by students.

Students and professionals performed similarly when compared for TDD because both the groups were novices in applying it. The further investigation through the second

---

7 We do not elaborate on the quality and productivity aspects of the used experiments because they are beyond the scope of this dissertation.
comparison (treatment-task [object] combination) suggested the possible effects of object complexity (task type) and the development approach. Therefore, the difference in code quality between the two groups may not be only attributed to the type of participants. The results of both comparisons also suggested the effect of the experience of the participants with the development approach. This study further concludes that participants should be characterised based on their experience or skill set rather than the setting (academia or industry) they belong to. However, the results of this study are confined to its context of investigation.

Article II contributes to the dissertation by supporting the use of students as proxies for novice professionals in the experimental investigations of Article III and Article IV. The unit testing context of this study makes it relevant to the software testing context of Article III and Article IV. However, the investigations in Article III and Article IV do not involve application/program development or interaction with any software to perform testing. This article answers RQ2 of this dissertation. The author of this dissertation was responsible for planning this empirical study, analysing the data of the experiments from the perspective of the study’s objective and reporting it.

4.3 Article III: a controlled experiment on time pressure and confirmation bias in functional software testing

The findings of Article I revealed that the investigation of time pressure was missing from the perspective of a potential antecedent to cognitive biases. The objective of Article III was to address this gap by executing an experiment to validate whether time pressure is an antecedent to confirmation bias in software testing. The theoretical support to the examination of this phenomenon was also acquired from the relevant psychology literature (see Section 3.1). Before validating time pressure’s postulate, the article also aimed to examine the manifestation of confirmation bias leading to confirmatory behaviour by software testers in software testing. The specific context of the investigations was test case designing for functional software testing. The experiment was executed in an academic setting in 2015 with 43 graduate students: 22 in the time pressure (TP) group and 21 in the no time pressure (NTP) group. A balanced design was achieved after data collection (see Chapter 3) because the data of one of the participants were dropped from the time pressure group.

Fig. 3 and 4 present the box plots of the data, i.e. the designed test cases in the TP and NTP groups, and of the rate of change measure. We can see in Fig. 3 that
the medians of consistent test cases in both groups are much higher than the medians of inconsistent test cases. However, the medians of the consistent test cases and the inconsistent test cases between the TP and NTP groups were near each other. The values of $z$ in the box plot for the rate of change also showed that all participants designed relatively more consistent test cases compared to inconsistent test cases in both the groups. It also depicts a minor difference between the TP and NTP groups as an effect of time pressure.

The results of the statistical significance tests showed that testers significantly manifested confirmation bias with a $p-value = 2.161e-08$ and an effect size $r = 0.598$, which is large. The study, however, could not provide evidence that time pressure promotes confirmation bias, neither in absolute terms of the designed test cases ($p-value = 1.001e - 1$) nor in terms of the rate of change $- z$ ($p-value = 8.72e - 1$). Although, the effect size ($d = -0.357$) was small to medium (for an effect of time pressure) when observed for $z$.

The results indicate a need to replicate the study with a large sample size to observe a possible effect of time pressure on confirmation bias. A sanity check performed to validate the time pressure construct revealed that the time pressure group perceived more time pressure compared to the participants in the no time pressure
Fig. 4. Article III: Rate of change (z).

group ($p\text{-value} = 1.231e-3$, $r = -0.469$ [medium to large]). This showed that we successfully operationalised time pressure. In addition to this, we also introduced new terminology to refer to the types of test cases, i.e. consistent and inconsistent (see Section 3.3.1). The terminology offers a new perspective on the (dis)confirmatory nature of test cases, especially when compared to earlier work, presented in Section 2.2. Our terminology offers straightforward comprehension of the types of test cases compared to the probable ambiguity in the understanding of positive/negative test case terminology.

Article III contributes to the dissertation by answering RQ3 and a part of RQ4, related to time pressure’s effect on confirmation bias. The study also contributed to the need for Article V by providing additional evidence in support for the existence of confirmation bias among software testers. Moreover, the result that time pressure could not promote confirmation bias served as a motivation for the investigation conducted in Article IV, which is explained in the subsequent section. The author of this dissertation was responsible for planning, executing and reporting this study.

4.4 Article IV: effect of time-pressure on perceived and actual performance in functional software testing

We could not find evidence in Article III that time pressure affects confirmation bias. It was, therefore, worthwhile to investigate the effect of time pressure on the actual and perceived performance (self-assessment) of software testers.
We replicated the experiment of Article III in 2016 with the same experimental settings. The objective was to compare performance as an effect of time pressure. The same experimental execution enabled us to use the data of the original experiment alongside the data from the replicated experiment, adding up to the data of 87 participants.

The results of the study showed that the actual performance (see Chapter 3 for the metric) of the participants in the time pressure groups significantly differed ($F[1, 84] = 8.523, p - value = 0.004$) with medium to large effect size ($\omega^2 = 0.079$) from the participants in the no time pressure groups. Fig. 5 presents the line error-bar graph of the actual performance. It can be seen that the means of the NTP groups were higher than the means of the TP groups. The perceived performance did not differ between the two groups, i.e. time pressure did not affect the perception of accomplishment (performance); $F(1, 84) = 0.210, p - value = 0.648$. The line error-bar graph of perceived performance (Fig. 6) shows less difference between the NTP and TP groups for the experimental

---

Republished with permission of Association for Computing Machinery, from Effect of time-pressure on perceived and actual performance in functional software testing, Salman & Turhan, 2018; permission conveyed through Copyright Clearance Center, Inc.
session E2 compared to a relatively larger difference for the experimental session E1. The statistical significance test, however, did not reveal a significant difference.

A significant negative correlation ($r = -0.284$ [small to medium], $p-value = 0.008$, $R^2 = 0.08$) existed between the actual and perceived performance when controlled for the effect of time pressure and experimental sessions (i.e. the original-2015, the replicated-2016). This explains the perceived performance accounted for 8% of variance in the actual performance. A sanity check performed in this study also confirmed that we successfully operationalised time pressure: $p-value << 0.000$, $r = -0.517$ [large].

Article IV contributes to the dissertation by answering RQ4. This provides empirical evidence in support of the effect of time pressure on actual performance with an indication that testers are unaware of the actual effect. The author of this dissertation was responsible for planning, executing and reporting this study.
4.5 Article V: what leads to a confirmatory or disconfirmatory behaviour of software testers?

The purpose of the study was to explore antecedents and to understand why and how they lead software testers to confirmation bias. The existing literature provides evidence of confirmation bias among software testers pertaining to certain factors (see Chapter 2). Article III also provided evidence in support of confirmation bias among software testers. The additional motivation for exploratory studies on cognitive biases in SE resulted from Article I. Moreover, Article III could not provide evidence that time pressure promotes confirmation bias, therefore, it was worthwhile to explore other antecedents to confirmation bias that may later be validated through experimentation. This study also intended to provide more insight into the results of Article III if time pressure was also identified as an antecedent to confirmation bias in software testing. In the context of this study, a confirmatory behaviour occurred when a tester designed/executed a consistent test case; similarly, a tester manifested disconfirmatory behaviour with the design/execution of an inconsistent test case. We applied the grounded theory approach to address the objective of our study, and we interviewed twelve software testers (by job role/title) to collect our data.

We identified twenty antecedents to (dis)confirmatory behaviour. These antecedents were categorised into nine categories. In addition to leading to a confirmatory or disconfirmatory behaviour, a few of them led to both behaviours. Both behaviours, i.e. confirmatory and disconfirmatory resulted in enhanced completeness of a test suite in terms of design and execution. In other words, the participants reported to design/execute both consistent and inconsistent test cases.

Nine out of the twenty antecedents lead to confirmatory behaviour. Medium or low priority of a functionality or use case, e.g. when functional priority is low, then, testers give less or no consideration to the design or execution of inconsistent test cases, which leads to a confirmatory behaviour. Furthermore, in case of a minor change to functionality or bug-fix, testers’ behaviour is confirmatory because of change’s minor impact on the system; the antecedent, is change/fix size. Incomplete requirements limit the testers to test what is minimally specified in the requirements, which leads to confirmatory behaviour. When testers adhere only to test the devices (e.g. iPad) prioritised by customers, as a result of customers’ test suite reviews (external party review), they manifest confirmatory behaviour. Moreover, automated test suite review is also an antecedent to confirmatory behaviour because it is performed by manual testers.
who lack expertise in advising for the handling/automation of complex inconsistent test cases. Automated testing is confirmatory because it is difficult to automate every inconsistent test case, and also the (automated) handling of unexpected output as a result of certain test cases’ execution. Another confirmatory antecedent is ambiguous requirements as testers are not able to completely understand them which impedes them from the designing of inconsistent test cases. Time pressure was also identified as a confirmatory antecedent because testers prefer to validate the behaviour of the system as specified in the requirements. In case of a high priority functionality, inconsistent test cases are also prioritised equally. However, it may not lead to complete test suite designing or execution due to time pressure.

We identified eight antecedents to disconfirmatory behaviour. It is promoted by detection of errors because testers exercise the system to find more errors. In other words, it promotes a code-breaking attitude among software testers. Additionally, this antecedent also leads to the addition of more inconsistent test cases in the suite. Absence of errors is also disconfirmatory because it impels testers to rethink their testing approach and their perspective towards the system under test (SUT). A software engineer who is a developer and tester is a disconfirmatory antecedent because testing as a tester enables a different and enhanced perspective to test the system. When a tester executes test cases that are previously designed or executed by another tester, it also leverages testing from an enhanced and different perspective. This practice acts as an antecedent to disconfirmatory behaviour, termed as complement testing. Disconfirmatory behaviour is also due to no time pressure because testers execute more inconsistent test cases. Furthermore, the past experience of testing that is irrespective of experience in any particular domain is disconfirmatory. Functional change requests lead to both behaviours for designing or execution of test cases. Clarifying (ambiguous) requirements is also an antecedent to both behaviours. Testing of a production bug fix, i.e. retesting a functionality due to a reported bug on production, also leads to both behaviours.

The antecedents, internal party review, i.e. review of manual test suites performed by team members, and project-specific testing experience not only lead to a disconfirmatory behaviour but also improve the completeness of a suite. For the antecedent, the change/fix size of the functionality, a major change leads to a disconfirmatory behaviour. Table 5 summarises why and how these antecedents lead to (dis)confirmatory behaviour with respect to their categories.

Fig. 7 presents an integrative diagram of our generated grounded theory. The figure explains the relationships among the categories, especially among the identified
antecedents. The antecedents belonging to their respective categories can also be seen in this figure, e.g. the antecedents past experience and project experience belong to the experience category. The letters, C, D, B, lB, before the antecedent names denote the type of the antecedent, e.g. automated testing is a confirmatory antecedent, project experience is disconfirmatory (D) but also (+) promotes both (B) behaviours and high priority is confirmatory but both behaviours is with limited test suite design or execution (lB). Time can be seen as one of the most connected categories to the others, which emphasises its influential role. Due to time pressure, testers perform high priority testing. Additionally, testers cannot perform exploratory testing that promotes disconfirmatory effect of the antecedent, detection of errors. Testers are also not able to clarify ambiguous requirements in a time-pressed situation. Automated testing aid in promoting the disconfirmatory behaviour under time pressure because it is possible to speedily (relative to manual testing) execute consistent test cases. This, in turn, makes it possible for the testers to test those inconsistent test cases that could not be automated. Time pressure is also another reason for confirmatory behaviour among testers when testing a minor functional change/fix.

The antecedents of the experience category play multiple roles. For example, testers based on their experience, execute either consistent test cases before inconsistent test cases or otherwise (specific behaviour) for the antecedents of the functionality retesting category and in a time pressure situation. Testers also define priorities (priority) of functionalities or scenarios based on their experience. Testers further manifest a specific behaviour (design/execution of inconsistent test cases before consistent test cases) due to experience as an effect of the antecedents of perspective change and functionality retesting categories. The third most connected category is, test suite reviews. Reviews by an external party define priorities (priority), e.g. platform (windows, android, iOS) priorities. Automated suite review practices contribute to the confirmatory nature of automated testing because of test automation skills limitations of manual testers.

Article-V contributes to the dissertation by answering RQ5. The generated theory explains the antecedents to (dis)confirmatory behaviour, why and how these behaviours occur among software testers, and how the antecedents are interrelated. The author of this dissertation was responsible for planning, executing and reporting the study.
Fig. 7. Article V: integrative diagram.
<table>
<thead>
<tr>
<th>Antecedent</th>
<th>Why &amp; How</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experience</strong></td>
<td><strong>Experience in testing enables how to approach the system, and which particular functionality checks to test for.</strong></td>
</tr>
<tr>
<td><strong>Project Experience</strong></td>
<td><strong>It enables testers to visualise the end-to-end functional flow of a system, and a good learning of the customer domain.</strong></td>
</tr>
<tr>
<td><strong>Priority (of a functionality, user stories, or platforms)</strong></td>
<td><strong>Consistent test cases have higher priority, but inconsistent test cases also acquire an equal priority in the case of high priority scenarios of a certain functionality.</strong></td>
</tr>
<tr>
<td><strong>Medium or Low Priority</strong></td>
<td><strong>Testers give either less or no consideration to the design or execution of inconsistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Ambiguous Requirements</strong></td>
<td><strong>They impede correct and complete test case designing, which results in the design/execution of mostly consistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Clarifying Requirements</strong></td>
<td><strong>Clarifying ambiguous requirements leads to the designing of both consistent and inconsistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Incomplete Requirements</strong></td>
<td><strong>Confirmatory because they limit the testers to test only what is minimally specified.</strong></td>
</tr>
<tr>
<td><strong>Production Bug Fix</strong></td>
<td><strong>Testing the fix first is confirmatory, which is followed by the testing of relevant inconsistent cases and other consistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Change Request</strong></td>
<td><strong>Both behaviours occur while designing/executing cases for a change request.</strong></td>
</tr>
<tr>
<td><strong>Change/Fix Size</strong></td>
<td><strong>A minor change is confirmatory due to its minor impact on the system. Both behaviours occur in the case of a major functional change.</strong></td>
</tr>
<tr>
<td><strong>Internal Party Review</strong></td>
<td><strong>When performed by members of the same project, it is disconfirmatory and also enhances the completeness of a suite.</strong></td>
</tr>
<tr>
<td><strong>External Party Review</strong></td>
<td><strong>It is performed by customers who define which devices to test, and set priorities. Therefore, only adhering to those priorities is confirmatory.</strong></td>
</tr>
<tr>
<td><strong>Automated Test Suite Review</strong></td>
<td><strong>Manual testers review to validate conformity with the manual suite. They are not experts in automation to assist with the handling and coverage of inconsistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Automated Testing</strong></td>
<td><strong>Confirmatory because it is difficult to automate every inconsistent test case and to handle unexpected results.</strong></td>
</tr>
<tr>
<td><strong>Detection of Errors</strong></td>
<td><strong>It leads to further investigation, e.g. through exploratory testing, and sometimes the addition of more inconsistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Absence of Errors</strong></td>
<td><strong>Disconfirmatory because it leads to rethinking the test approach and assessing a test suite from a different perspective.</strong></td>
</tr>
<tr>
<td><strong>Time Pressure</strong></td>
<td><strong>Consistent test cases are prioritised because they ensure the behaviour of the SUT per the documented specifications.</strong></td>
</tr>
<tr>
<td><strong>No Time Pressure</strong></td>
<td><strong>This leads to more testing, e.g. through exploratory testing and the execution of more inconsistent test cases.</strong></td>
</tr>
<tr>
<td><strong>Developer &amp; Tester</strong></td>
<td><strong>Disconfirmatory because testing as a tester, compared to development, changes and broadens perspective for the SUT.</strong></td>
</tr>
<tr>
<td><strong>Complement Testing</strong></td>
<td><strong>A tester executing test cases that were previously executed or designed by another tester promotes disconfirmatory behaviour.</strong></td>
</tr>
</tbody>
</table>
5 Discussion

This chapter answers the research questions of this dissertation based on the findings of the original articles. The answers are discussed in the specific context of the questions, i.e. a comprehensive discussion on the results of the original articles can be found in the respective publications. The chapter, in the end, discusses the threats to the validity of the results of this dissertation. Similar to the previous section, detailed threats to validity are present in the respective publications of the original articles.

5.1 RQ1: what is the state of the art research on cognitive biases in software engineering (SE)?

**Answer:** We identified 65 articles from 1990 to 2016 that focused on the antecedents, effects, or debiasing techniques of 37 cognitive biases in SE. Anchoring bias, confirmation bias and overconfidence bias were the most investigated cognitive biases. Most cognitive biases were studied from the antecedents and effects perspectives. Experiment was the leading research method in this research paradigm. The research on the subject area mostly appears in journals or academic magazines, and most of the work has been performed in software engineering management and software construction areas of SE. However, the research has been disconnected, i.e. it has been lacking in making references to the other investigated, relevant cognitive biases.

**Discussion:** The findings highlighted the need for more qualitative and multimethodological research because research on cognitive biases is concentrated mostly on establishing causality relationships, e.g. anchoring and adjustment bias effects on effort estimation (van Vliet & Tang, 2016). Qualitative and multimethodological research could help establish the relative importance of cognitive biases from the perspective of the intensity of their effects on different stages of software development. It would also help in understanding the underlying mechanisms of the manifestation of cognitive biases in the SE industry.

Some knowledge areas of SE (per SWEBOK) are completely ignored within the investigation of cognitive biases, e.g. configuration. Similarly, several bias categories and biases, e.g. the ambiguity effect, that are investigated in information systems should also be investigated in SE. Since Fleischmann et al. (2014) identified cognitive biases in information systems for the areas of software development and application systems,
which are also software engineering domains, the findings also suggest focussing more on proposing debiasing techniques because, compared to the research on effects and antecedents to biases, this area lacks attention.

We also found that a few biases have inconsistent terminology issues in the identified literature, e.g. confidence bias has also been found as overconfidence bias. It was also observed that some apparently different biases are alike because they simply occur in different circumstances but essentially are based on a similar cognitive process. For example, preoccupation with a presentation is the framing effect (Stanovich E., 2009), but preoccupation with a number is known as anchoring bias (Parsons & Saunders, 2004). The contexts in which the concepts emerged in social science are different from the contexts in which they have been employed in applied sciences. This miscommunication across the disciplines has led to certain confusions in SE research, e.g. confusing heuristics with biases; anchoring and adjustment is a heuristic that leads to adjustment bias. The disconnection among the studies and inconsistent terminology issues were impediments towards synthesising the research on cognitive biases in SE.

5.2 RQ2: are students representative of professionals in software testing experiments?

Answer: The findings suggest that students may be recruited to represent novice professionals in software testing experiments. In the context of the findings, the term novice refers to those professionals who are either novice in the industry or are a novice concerning the use of a technique or technology. In other words, experimental participants might be categorised based on their experience rather than the setting (industry, academia) they belong to. The findings of this question are in the context of development practices, therefore, students may represent novice professionals for the experiments that aim to investigate new techniques or technology. The findings are confined to the context of the experimental investigation, which is explained further. Moreover, a single empirical study is not sufficient to answer this question.

Discussion: The findings cannot be generalised to all contexts in the SE discipline. The analysis of the values of the internal code quality metrics suggests the possible effect of other factors on the findings apart from the types of the participants’ groups. When students and professionals were compared for TLD, professionals produced a better quality code. We found that this could be attributed to the experience of professionals in
programming and with the development approach, i.e. TLD because, when they were compared for TDD, which was new to both of the groups, the code quality was similar.

Further investigations revealed that the code quality might also be associated with the complexity of the experimental objects and the particular development approach, i.e. TDD or TLD. This effect could have also led to the observed differences between professionals and students. The mentioned potentially influencing factors were not a part of the theoretical construct; if considered, they may reveal an interaction with the internal code quality.

The results of this study can be compared with McMeekin et al. (2009), who also observed professionals’ performance as better than students’ in defect detection due to their experience level. The findings are also similar to Höst et al. (2000) and Svahnberg et al. (2008), in which students and professionals performed similarly. However, in contrast to our study, in which we assessed technical outcomes (programming tasks), they assessed non-technical outcomes rather than subjective opinions, e.g. conceptions of factors (Höst et al., 2000) and perceptions of requirements’ selections (Svahnberg et al., 2008). Moreover, our study is contextualised within software testing. The results show that similarity between students and professionals may vary with the SE area, and they are also dependent on the experience of professionals with the assessed approach, e.g. TLD in our case.

5.3 RQ3: do software testers manifest confirmation bias in software testing?

Answer: The findings of Article III showed that the participants manifested confirmation bias by designing more consistent test cases compared to inconsistent test cases for functional software testing. The results not only held for the pool of all participants but also when observed the participants separately in the no time pressure group and the time pressure group because this was investigated with the theoretical construct designed for observing time pressure’s effect on confirmation bias.

Discussion: This observation is confined to multiple factors of the experimental investigation. The experimental object was a realistic one, the experimental participants were inexperienced in software testing, and the participants were unfamiliar with the domain. The presence of these factors could have made the designing of inconsistent test cases difficult compared to consistent test cases. For example, (higher) experience in development or testing was observed to lower confirmation bias levels but not
to eliminate it (Calikli & Bener, 2013; Leventhal et al., 1994; Teasley et al., 1994). Additionally, the participants did not interact with the running application, rather, they were limited to the designing of test cases only. According to Teasley et al. (1994), error feedback of the application, which could occur on interaction, might lead them to design more inconsistent test cases. However, the absence of these mentioned factors from our experimental investigation may not eliminate confirmation bias, albeit minimise it. Per the findings of Teasley et al. (1994), the tendency for using positive test cases was more than negative test cases even at the higher levels of expertise and complete requirements specification.

Our answer to this question supports the findings of Teasley et al. (1994), Leventhal et al. (1994) and Causevic et al. (2013) that the participants manifest confirmation bias when they design test cases. However, at the same time, our results also challenge the findings of these studies from the perspective of the measurement of confirmation bias. Leventhal et al. (1994) and Teasley et al. (1994) assessed confirmation bias in terms of valid (positive) and invalid (negative) equivalence classes. Therefore, their results might differ when their data is assessed from the perspective of (in)consistent terminology. In our case, we did not consider a test case inconsistent if it used the data from an invalid equivalence class. Instead, it was considered a consistent test case if the invalid behaviour of the program/application was specified in the requirements. Causevic et al. (2013) reported having used a “built-in mechanism” to distinguish between positive and negative test cases, hence, it is unclear how it was actually carried out.

5.4 RQ4: how does time pressure affect confirmation bias as well as the actual and perceived performance of testers?

Answer: We could not observe time pressure as promoting confirmation bias. Time pressure significantly affected actual performance, which was measured in terms of percentage coverage given to the specifications. However, time pressure was not observed to affect the perception of performance among the participants. The participants’ self-assessment of the coverage given to the specifications was sustained irrespective of time pressure.

Discussion: Article III could not provide statistically significant evidence of time pressure’s effect. The effect sizes of the absolute count hypothesis test (large) and the rate of change ($z$) hypothesis test (small) indicated that time pressure may affect confirmation bias. The reasons for these findings are manifold. The experiment (Article
III) used a realistic experimental object, which made the task of designing test cases complex for novice professionals (students) despite the (in)consistent type of test cases. Additionally, unfamiliarity with the domain of the experimental object further increased the complexity of the task.

The (limited) experience of the participants could not let them design a considerable amount of inconsistent test cases. This limited the no time pressure group as well in designing a considerable amount of inconsistent test cases that could have otherwise led to the observation of time pressure’s effect on confirmation bias. The following investigation, Article IV, showed that time pressure affects the percentage coverage of the specifications, irrespective of the types of the designed test cases. As explained earlier in Chapter 3, we replicated the experiment that was used to answer the effect of time pressure on confirmation bias to answer the performance part of RQ4. We, therefore, assessed the same data but used different metrics to measure the (actual and perceived) performance.

The findings of Article IV also showed that participants overestimated their performance when self-assessed compared to their actual performance. In this context, the tendency to overestimate one’s accuracy regarding the coverage given to specifications may indicate the manifestation of overconfidence bias. The data suggests that the experience of the participants might also affect the accurate self-assessment of the performance because the participants of the original experimental execution assessed their performance relatively more accurately compared to the participants of the reproduced experiment. This may imply software test engineers with less experience are more susceptible to overconfidence bias. We did not account for examining the effect of experience (Article IV), otherwise, it could have an interaction with time pressure in determining its effect on perceived (self-assessed) performance. The negative correlation between the actual and perceived performance, when controlled for the effects of time pressure and experimental sessions, could be attributed to perceived performance in E2 (Fig. 6). The data shows that the participants of E2 perceived performing better under time pressure compared to the no time pressure group.

The findings of this question cover the limitations of existing research highlighted in Chapter 2. Therefore, the findings of this question are not directly comparable with any study from the known SE literature. We, however, position the findings from other aspects. For example, from the perspective of time pressure affecting the dependent

---

9(Over)confidence bias: “The tendency to overestimate one’s own skill, accuracy and control over one’s self and environment” (Mohanani et al., 2018, p. 21).
variable (confirmation bias, in our case), our findings are similar to Topi et al. (2005), Nan and Harter (2009) and Mäntylä et al. (2014) because, in these studies, time pressure also could not affect the dependent variables (see Section 2.3). Our finding that time pressure affected the actual performance is in contrast to the finding of Mäntylä et al. (2014). Their study did not provide evidence of time pressure’s effect on effectiveness - measured as the defect count and test case score (Mäntylä et al., 2014).

Based on the findings of RQ2, we recruited students as proxies for novice professionals to answer RQ3 and RQ4. However, there was a difference between the nature of the tasks, i.e. the way they were performed, between Article II and Article III, Article IV. Articles III and IV used pen-and-paper task for the experimentation, whereas, the experiment used for the empirical study of Article II used the programming tasks, which required the use of the software development tools. The focus of the studies in Articles III and IV was to design the test cases instead of test cases’ execution. This is similar to designing manual test cases in the industry, either they are designed in a simple application (e.g. excel sheets) or a designated tool that supports test cases’ execution. Therefore, the difference in the nature of tasks cannot hinder the transferability of the results of Article II, i.e. the use of students to represent novice professionals from this aspect.

5.5 RQ5: why do software testers manifest confirmation bias?

Answer: We identified nine antecedents to confirmation bias. They mainly belong to certain phases of the testing process, e.g. test case designing, test suite reviews, test executions. Confirmation bias occurs when ambiguous or incomplete requirements are used to design test cases. Testers are not able to understand such requirements well or completely, which limits the design of inconsistent test cases. The priorities of functionalities or user stories also act as an antecedent. For a medium or low priority, testers either design/execute little or no inconsistent test cases. In the case of a high priority, testers design both types of test cases, but test suite completeness may be limited in design/execution due to time pressure. Similarly, when a functionality change/fix size is minor, testing is confirmatory because of minor impacts on the SUT. In the case of external reviews of test suites - performed by customers - testers only adhere to customers’ preferences, which limits the complete test suite execution. Automated test suite reviews are not performed by automation experts, which also limits the
accommodation of difficult-to-automate inconsistent test cases. Due to this, *automated testing* is mainly based on the design/execution of consistent test cases.

Lastly, the antecedent from organisational environment is *time pressure*. Testers under time pressure prefer to execute consistent test cases because they validate the behaviour of the SUT with respect to the documented specifications. A similar preference is also made for other activities under time pressure, e.g. in the case of *ambiguous requirements*, testers are unable to clarify the requirements. This makes time pressure a major confirmatory antecedent that also limits the disconfirmatory effect of the identified disconfirmatory antecedents, e.g. clarifying requirements (see Section 4.5).

**Discussion:** In general, software testers manifest confirmatory behaviour before disconfirmatory behaviour. In other words, the design/execution of consistent test cases is preferred over inconsistent test cases. However, this sequence may change for some testers because of their experience, personality, or nature. The occurrence of disconfirmatory behaviour before confirmatory behaviour may mitigate confirmation bias but may not assure a defect-free SUT. For example, in the case of executing inconsistent test cases before consistent ones, time pressure may limit the complete execution of consistent test cases that may also reveal errors.

In comparison to the existing literature on confirmation bias in software testing, we identified thirteen new antecedents to (dis)confirmatory behaviour: *high priority, medium or low priority, ambiguous requirements, production bug fix, change request, change/fix size, internal party review, external party review, automated test suite review, free time, automated testing, developer and tester and functionality switching*. There were seven antecedents that have been studied in the existing literature, but our study could not identify them. The main reason for this could be that our grounded theory was based on the data of software testers who were employees of the same company. Moreover, almost all the testers worked in agile (scrum) projects, which limited our observation of the possible effects of company culture, company size, job titles and development methods (Calikli & Bener, 2010, 2014; Calikli et al., 2010).

The time category emerged as one of the major categories in the grounded theory due to its antecedent, time pressure. Our findings, particularly the effects of time pressure, are similar to the findings of Wilson and Hall (1998), Baddoo and Hall (2003), Cataldo (2010) and Shah et al. (2014). These qualitative studies also identified time pressure as a factor with negative effects in their respective studied contexts. Time pressure also limits the design/execution completeness of a test suite, per the grounded theory. This qualitative finding supports the finding of the role of time pressure regarding actual
performance (Article IV); time pressure significantly affected the actual performance of the participants. In Article IV, the actual performance (percentage coverage given to the specifications) was less in the TP groups compared to the NTP groups.

Our experimental study (Article III) could not find time pressure to significantly affect confirmation bias. Among the multiple possibilities discussed for the experimental study results, one factor was the experience of the participants. Experience emerged as one of the major categories in the grounded theory (Article V). The antecedents of this category, past experience and project experience, led to disconfirmatory behaviour among software testers, i.e. promoting the designing of inconsistent test cases. Therefore, from this perspective, if the participants of Article III were experienced, then, they could have designed relatively more inconsistent test cases irrespective of the TP or NTP groups. However, to observe the effect of time pressure on confirmation bias with experienced participants, duration less than 30 min may need to be used for the TP group. In other words, experiment with experienced testers (e.g. practitioners) may require different durations to properly operationalise the time pressure construct.

5.6 Threats to validity

We structure this section based on the types of validity threats and discuss the findings in the respective context. We follow the classification and guidelines on threats proposed by Wohlin et al. (2012) and Zhou, Jin, Zhang, Li, and Huang (2016).

5.6.1 Construct validity

The results of our SMS revealed inconsistent terminology issues regarding the names of cognitive biases in the SE literature. It made analysis difficult because we were not sure if we interpreted it in the intended manner. In the grounded theory study, to avoid misunderstanding the data, we shared the summaries of the interviews with the participants for their confirmation. We also eliminated the theoretically conflicting and self-conflicting data from the interviews. To be sure to collect the right data using interviews, we developed an interview script that we piloted before conducting actual interviews.

10Theoretically conflicting data refers to data that indicated a conflict between interviewee’s understanding of the term and its theoretical definition.
The time pressure construct studies were prone to mono operation bias threat because they involved only one experimental object (MusicFone) - the interaction of task and treatment. The RQ2 study was not prone to this threat because it used multiple experimental objects. The mono-method bias threat in the time pressure construct studies was resolved by executing a pilot run and calculating an inter-rater agreement for the classification of test cases as inconsistent and resolving confusions using a discussion session. The students vs professionals study used a tool (see Section 3.3.2) to extract the code quality attributes and we quantified internal code quality using twenty code attributes. It, therefore, addressed the mono-method bias threat in that respective study. We addressed the threat of testing and treatment interactions in Article III and Article IV by instructing the participants to perform the experimental task as a regular class activity. The participants were also unaware of the treatment of the studies.

A possible threat that is associated with Article III study is the total count of the inconsistent test cases \((IC)\). The validity improvement step, as mentioned earlier in Chapter 3 and Section 3.3.1, increased the count of \(IC\) test cases in the heuristic baseline after the data collection step. If this study (Article III) is replicated for the investigation of the same theoretical construct, the count of \(IC\) test cases may increase to a further higher value. The consequence of this might be the observation of results that would be more in favour of (the manifestation of) confirmation bias.

5.6.2 Internal validity

The mapping study was prone to publication bias threats for multiple reasons, e.g. the studies may not be indexed in the databases that we searched, the studies may be available in domains other than ‘computer science’ because we limited our search to ‘computer science’ or similar domains while searching for the literature in databases. Cognitive biases have been studied in isolation per the findings (Section 4.1), which impeded the synthesis of the research on the topic.

In the GT study, we performed a pilot coding of a randomly chosen interview from the set of interviews. The coders of the pilot coding used a list of terms to identify the antecedents to (dis)confirmatory behaviour, which was prepared by the interviewer. Therefore, the pilot coding may have been prone to researcher bias. However, the low values of inter-rater agreement do not support the probable effect of researcher bias.

We discuss internal validity threats that belong to multiple group threats and social threats for the studies that involved experimentation (Wohlin et al., 2012). To address
interaction with selection threats for the studies that operationalised time pressure constructs, all the participants were trained together and were randomly assigned to treatment and control groups. The joint training sessions also prevented compensatory rivalry or resentful demoralisation threats. The students vs professionals study could have been prone to selection-maturation interaction because professionals, based on their programming experience, might have learned the concept and application of TDD more quickly compared to students during the training phases. The results of the RQ2 study do not suggest the effect of selection-maturation interaction. Additionally, applying TDD may require a steep learning curve, which further alleviates the possibility of selection-maturation interaction threats (Turhan et al., 2010).

The studies of time pressure constructs were not prone to the imitation of treatments threat because the treatments were applied in parallel, in different rooms, to the experimental and control groups. The RQ2 study is also not prone to this threat because it was a within-subjects experimental design, and the control and treatment sessions took place on separate consecutive days - see Section 3.3.4. None of the groups, i.e. experimental and control, in any of the experimental studies was compensated in a special way, which prevented compensatory equalisation of the treatment threat. For the time pressure construct studies, an incentive in the form of bonus marks was announced to encourage the students to participate in the experiment. We do not presume it as a threat to the internal validity because the incentive neither constituted undue influence nor was it coercive (Commission, 2012).

### 5.6.3 External validity

In order to increase the external validity of our mapping study, we performed several strategies. For example, we performed iterative backward snowballing on the reference list of the primary studies, analysed the profiles of prolific authors in the subject area to find more potential studies and examined the first 150 results of Google Scholar. The generated grounded theory (Article V) may not work for other settings because the theory is based on the data of software testers who were employees of the same company.

The findings of Article II were less prone to the interaction of selection and treatment threat. To compare students and professionals, the samples were chosen from the intended populations, i.e. students from the academic environment and professionals from the industrial environment - see Section 3.3.3. Both sample groups were heterogeneous
in terms of their experience regarding programming skills, programming tools and years of experience. Based on the findings of Article II, we used students as proxies for novice professionals in the rest of the experimental studies. However, the aim was not to investigate the use of a new technique or technology but rather a cognitive approach and the effects of time pressure. Additionally, the student participants were first-year graduate students with a minor industrial experience.

The study comparing students and professionals observed the two groups in their respective environments using the tools that were used in the industry. The implemented experimental objects were both toy and realistic; the effects of these have already been discussed in Section 5.2. Therefore, the Article II’s study was not prone to the interaction of setting and treatment threat. The time pressure construct studies (Article III and Article IV) were also not susceptible to this threat. Although these studies used pen and paper to perform the tasks, the investigation focused on the designing of test cases instead of the test cases’ execution. In the industry, software testers (by job role/title) use requirements specifications to design test cases before test case execution for system, integration and acceptance testings. The object used in these studies is also a realistic object. Moreover, operationalising the time pressure construct (see Section 3.3.1) to observe its actual effects, the industrial environment was unsuitable because of uncontrollable noise. None of our studies were prone to the interaction of history and treatment threat.

5.6.4 Conclusion validity

The non-significant results of time pressure’s effect on confirmation bias, both in terms of absolute comparison and relative terms (rate of change), may be due to the type-II-error. This requires replications of the same theoretical construct with the larger sample size because the effect sizes indicated of a possible effect.

Before running the significance tests, we tested them for their assumptions - see Section 3.3.5. In the case of not meeting the assumptions, we chose the appropriate tests. This way, we prevented the threat of violated assumptions of statistical tests. We addressed the error rate threat, wherever needed, by applying the Bonferroni corrections to the significance levels. However, the results of students vs professionals' construct may have accumulated the type-I-error because we did not apply the Bonferroni corrections. The Bonferroni corrections, if applied, could have introduced the type-II-error to our conclusions for the respective construct.
We executed a pilot run before the actual experimental execution of time pressure and confirmation bias construct, which addressed the reliability of measures threat in the studies that investigated the effects of time pressure. The computing of an inter-rater agreement for the categorisation of test cases as *consistent* or *inconsistent* also further addressed the reliability of measures threat. We prepared a scripted guideline for the experimenters to apply the treatment and control of the time pressure construct in a uniform and consistent manner. We also formulated and tested a sanity check hypothesis to validate the operationalisation of time pressure. These two steps addressed the threat of reliability of the treatment. The experimental executions performed in the academic environment enabled control over random irrelevancies that might have otherwise influenced the results.

We performed several steps to enhance the intercoder reliability and intercoder agreement in the GT study. In addition to the pilot coding step that involved four coders, the main coder, coded the other interviews in iterations. We held joint discussion sessions after every iteration to resolve confusions in coding. The generated grounded theory was prone to threats because we did not perform theoretical sampling. Theoretical sampling refers to further data collection to saturate certain concepts or fill gaps in the emerging theory (Stol et al., 2016).
6 Conclusions

We, once again, state the objective of this dissertation: to examine the manifestation of confirmation bias, why and how it occurs among software testers, the role of time pressure in confirmation bias’ promotion, and the perceived role of time pressure.

This chapter concludes the work with respect to the objective. Additionally, we list our contributions to the body of SE knowledge, as per that made by the research questions with the help of the original research articles. We also present implications of this work for the industry, academia, and research followed by the possibilities to extend this work.

6.1 Contributions

In order to address the objective of this dissertation, the research was conducted in two phases, Phase 1 and Phase 2. We first present the contributions made to the body of SE knowledge by the studies of Phase I, followed by the contributions of the studies of Phase 2. The systematic mapping study contributes:

(a) A summarised review of the state of the art research on cognitive biases in SE.
(b) Implications and future research directions based on the current state of the research.

The empirical study validating the theoretical construct of students vs professionals has made multiple contributions:

(c) Empirical evidence of the similar performance (code quality) of professionals and students for the development practice with which professionals were inexperienced.
(d) Empirical evidence that professionals’ performance differed from students for the development practice in which they were more experienced compared to students.
(e) Empirical evidence, based on objective measures (using Prest - see Section 3.3.2), that helps determine the possible use of students as experimental subjects in place of novice professionals.

We examined the manifestation of confirmation bias, why and how it occurs among software testers both quantitatively and qualitatively by using experimentation and grounded theory as research methods. These studies made the following contributions:
The experiment on confirmation bias provided evidence of its manifestation by software testers while designing functional test cases.

The experimental study also introduced a new perspective to refer to the types of test cases, a consistent test case and an inconsistent test case, in functional software testing in comparison to the existing work.

The grounded theory postulated nine antecedents to confirmation bias among software testers.

The theory also postulated antecedents that may mitigate confirmation bias - disconfirmatory antecedents.

The theory further explained how confirmation bias occurs, i.e. how the identified antecedents led to confirmation bias and the relationship between the antecedents.

The research work explains the role of time pressure in confirmation bias promotion also using two research methods. The studies applying these research methods contributed the following:

The experimental design that studied the effect of time pressure on confirmation bias in the software testing context is a novelty in SE, to the best of our knowledge.

The results of the above experimental execution revealed that time pressure could not significantly promote confirmation bias among software testers while designing functional test cases.

However, the category of time emerged as a major entity in our grounded theory. Its antecedent, time pressure, led software testers to confirmatory behaviour. Furthermore, time pressure either contributed to the confirmatory nature of other identified antecedents or reduced the disconfirmatory effect of the disconfirmatory antecedents.

The perceived role of time pressure was examined through the experimental study.

Empirical evidence that the perception of performance (accomplishment) sustained irrespective of time pressure.

Whereas, time pressure significantly affected the actual performance of the testers.

The observations regarding the perception of performance also indicated a possible occurrence of confidence bias among the testers.

This experiment was the first of its kind in a software testing context that compared actual performance with the perceived performance that was collected after the execution of the task.
Fig. 8. Mapping of RGs (research gaps), RQs (research questions), original articles and contributions.

Fig. 8 extends Fig. 1 (Chapter 1) by adding contributions mapping to it. The figure shows which articles made the above listed contributions, e.g. the contributions $f, g, k, l$ were made by Article III (A III).

### 6.2 Relevance to the industry

The contributions made by this work have several implications for practitioners. On an individual level, software testers should develop self-awareness for confirmation bias. It may enable them to mitigate the deleterious effects of confirmation bias on testing by reflecting their own actions regarding testing. Software testers need to develop a disconfirmatory attitude, especially if they are in the beginning of their career because, later in their career, experience may play a disconfirmatory role. On the managerial level, there is a need to promote and implement practices that may mitigate confirmation bias. For example, test reviews by manual testers help promote the disconfirmatory behaviour among software testers, per the results of the grounded theory. The manual testers should belong to the same project and should ideally be more experienced compared to the tester whose suite is under review. Automation test experts should review the automated test suites to help junior automation test engineers in the implementation of complex inconsistent test cases. It may not help with the technical limitations of the automated testing tools, e.g. handling of unexpected results, albeit, it may enhance the completeness of inconsistent test cases in the automated suite.
Time pressure emerged as one of the primary antecedents in grounded theory for confirmation bias. Automated testing, despite its relative confirmatory nature, may aid in performing effective testing in a time-pressured situation. To achieve this, manual and automation test engineers should test in collaboration. Manual testers can test those functionalities that are not yet automated or those sets of test cases that cannot be automated while automation engineers in the parallel test the other parts of the system. However, this strategy poses a challenge in having automated suites available for the entire project/system and always updated. It is important to have similar strategies in place to manoeuvre time-pressured situations because the disconfirmatory attitude of a tester alone may not ensure quality testing.

Software testers also need to improve their self-assessment skills. It would enable them to reflect well on their performance. It is important because the wrong perception may cause over or underestimation of their actual performance. Overestimation is particularly risky for the quality of testing irrespective of time pressure situations. It may increase feedback cycles between developers and testers and decrease external quality. Improvements in self-assessment of their performance may give them a chance not only to improve their actual testing performance but also an increased awareness of their technical strengths, both under time-pressured and no-time-pressured situations. It may also enable the testers to provide better estimates for their testing tasks during the planning of a task/project.

6.3 Relevance to academia and research

The work provides evidence that researchers may recruit students as representatives of novice professionals, especially when they intend to assess a new technology or, e.g. a new development practice. It may enable them to assess its immediate effects before introducing it to the industry. However, a single experimental study to assess new technology and introduce it to the industry is not sufficient from the scientific perspective.

Software testing courses at the universities should also teach students how to develop a disconfirmatory attitude. In addition to teaching the valid/invalid test case concept of functional testing, it should be encouraged to think outside-the-box for the requirements specification in hand. In other words, the concepts of consistent and inconsistent should also be introduced to the students. This may train them to assess the completeness and
correctness of the specifications from the perspective of implementing and testing the program.

### 6.4 Future research

This research proposes multiple future research opportunities. The results of the study of students vs professionals construct are confined to the study’s context. To enhance the external validity of the results, more experiments are required for different subject areas of SE. The experimental results of time pressure’s effect on confirmation bias suggest a need to replicate the experiment with a larger sample size. It may provide statistically significant evidence for time pressure’s effect on confirmation bias. This experiment, run with professionals with considerable years of experience, may validate the propositions concerning the *experience* category of the grounded theory. Also, an experimental study examining the effect of varying levels of experience and varied types of experience (e.g. project experience, testing experience in general) would enable understanding of which type or level is more alleviating for confirmation bias. There is also a need to study the interaction of task complexity and time pressure on confirmation bias. Introducing experience as a confounding factor in the design of perceived and actual performance experiment may reveal whether experienced testers perceive their performance more accurately.

The grounded theory was grounded in the data of the testers of only one company. Analyses of the data from the testers of other companies may modify it and make it more comprehensive. The antecedents that are particularly relevant to the characteristics of a company (e.g. company size, organisational culture) may also emerge in theory. Experimental studies may be conducted to validate the propositions put forth by the grounded theory, especially to establish the relative importance of the identified antecedents to (dis)confirmatory behaviour. The grounded theory research method can also reveal the antecedents to other cognitive biases and help to understand the underlying phenomenon.
References


van Vliet, H., & Tang, A. (2016). Decision making in software architecture. *Journal of


Original publications


Reprinted with permission from The Institute of Electrical and Electronics Engineers, IEEE (I, II and V), Springer [Open Access Article] (III), and Association for Computing Machinery, ACM (IV).

Original publications are not included in the electronic version of the dissertation.
721. Giunti, Guido (2018) 3MD for chronic conditions: a model for motivational mHealth design
727. Hurskainen, Sonja (2018) The roles of individual demographic history and environmental conditions in the performance and conservation of northern orchids
728. Haapalahti, Reijo (2019) Yksilön toimien vaikutukset aluekehitykseen: ammatilliseen perustutkintokoulutukseen liittyvät odotukset ja tulokset Pohjois-Pohjanmaalla
729. Tokkonen, Helena (2019) Say, Do, Make! : user involvement in information systems design
731. Mylonopoulou, Vasiliki (2019) MAD: designing social comparison features in health behaviour change technological interventions
732. Shevchuk, Nataliya (2019) Application of persuasive systems design for adopting green information systems and technologies
733. Tripathi, Nirnaya (2019) Initial minimum viable product development in software startups: A startup ecosystem perspective
THE EFFECTS OF CONFIRMATION BIAS AND TIME PRESSURE IN SOFTWARE TESTING

Iflaah Salman