Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Experimental investigation of the consequences of expected source code understandability

Lasse Merz

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Marvin Wyrich, M.Sc.
Commenced:	March 2, 2020
Completed:	October 27, 2020

Abstract

Understanding program code represents an essential part of most developers' work. Any maintenance task requires the comprehension of the corresponding code as a first step. For that reason, software companies pay close attention to the quality of their codebase. It has become a standard to incorporate static analysis tools in the software process in order to automatically identify code smells and help developers to improve their code. However, the majority of metrics that are used in static analysis tools lack empirical evidence. We do not know how these unvalidated metrics influence the cognitive process of developers in regard to program comprehension.

In this work, we investigate the consequences of presenting different understandability values to developers prior to them inspecting a code snippet. We analyze to what extent this understandability metric impacts the expectations, motivation, and affective states of programmers. To this end, we conduct an experiment through an online survey with 81 developers randomly assigned to one of three treatment groups with different presented understandability values. Before and after the task of judging the understandability of a code snippet, participants have to report their expectations, motivation, and affective state with regard to understanding the code snippet. In addition, two code snippets are used to evaluate differences in perception, motivation, affect, and understandability judgment as a result of the actual difficulty of the code snippets.

Our findings show no significant effect for expectations, motivation nor affective states as a consequence of the presented understandability value. However, we observe a significant positive linear relationship with expectations explaining 18.3% of the variance of motivation at an alpha level of 0.0056 with a large effect. Similarly, differences between expectations of understanding the code snippet before seeing it and the perception of understanding it afterwards demonstrate the same significant positive relationship with motivation difference. Our results show an even larger correlation between expectation to perception difference with a happiness difference of participants, indicating that being positively surprised by understanding a code snippet corresponds with increased motivation and happiness. Lastly, presenting programmers' different understandability values does not influence the assessment of the code snippet.

Generalization of these results is limited by the use of small code snippets of 20 to 30 lines of code. Furthermore, expectation and motivation are measured through a self-created and therefore unvalidated instrument.

The results showcase the importance of managing expectations in order to increase motivation and affect of developers. Additionally, contrasting to prior work understandability metrics seem to not pose a threat of biasing programmers in their expectations towards and assessment of source code.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Maintenance efforts comprise the main part of software development. Up to 70% of the costs in the life-cycle of a software product are spend on maintenance [BB01; Erl00]. It is important to preserve a clean and understandable codebase. Any maintenance task requires the comprehension of the artifact as a first step [Ray91; Rug00]. Therefore, understandability of source code is essential for all software maintenance activities [TN14]. Software developers have to understand the code in order to enhance performance, add new features, fix bugs, or perform refactorings [VVH97]. For that reason, understanding source code is an integral part of the work of most software developers. Professional programmers spend between 58% and 80% of their time on activities related to comprehending code [MML15; Tia11; XBL+18].

As a result, there exists a need for tools that help developers in this endeavor. Static analysis tools use rules, heuristics, and software metrics in order to identify potential problems, code smells, and bugs in the source code. Therefore, they serve to help developers in improving their code. They are an established part of integrated development environments in many software projects and big tech companies [SAE+18; VPP+19]. However, studies continuously show various problems with the metrics used in static analysis tools. These metrics are oftentimes based on flawed models, lack empirical validation, and are used beyond their intended context [SI94], produce many false positives [WAB09], and do not represent underlying software quality characteristics as they are understood by developers [PLB18]. Overall, static analysis tools only support few verified metrics and instead provide results based on an overwhelming amount of unvalidated metrics [NAG19]. Especially for code understandability, no empirical evidence exists for the validity of almost all proposed metrics [CSLB19; SBV+17].

Distinct cognitive processes are involved in program understanding which activates areas of working memory, attention, and language comprehension [PSA+20]. Therefore, we are interested in the extent to which unvalidated understandability metrics influence the cognitive processes of software developers. Many cognitive biases are recognized in different areas of software engineering and shown to negatively impact programmers' performance [CNA+20; MST+18]. To our knowledge, only one prior experiment specifically investigated the impact of understandability metrics in this context. It showed, that presenting different understandability metrics to developers about an upcoming code snippet significantly impacts their subsequent subjective understandability assessment of this code snippet [Pre20]. Hence, we aim to further explore more fundamental mechanisms involved in developers' interaction with code understandability metrics.

Research concerned with the placebo effect showcases the impact suggestions can have on the outcomes of people in terms of pain relief and many other clinical conditions [PFB08] as well as cognitive performance [TBG+18]. With regard to the underlying mechanisms, researchers identified expectations elicited through verbal suggestions and social information as part of the treatment to be an important factor in the explanation of placebo effects [SP04; SPB16; WA15]. These expectations then influence the motivation and affective states - emotions and moods - of people which in the end impacts their physiology, behavior, and reported experiences [WA15]. We believe, that values from understandability metrics act as suggestions for the comprehensibility of the corresponding code. Therefore, we analyze the expectations evoked through different code understandability values and how they subsequently influence motivation and affective states of developers in regard to understanding code.

As a result, we intend to gain a deeper understanding of the effects understandability metrics have on programmers. Thereby identifying factors to potentially improve such tools and acquiring new knowledge about the cognitive process involved in understanding source code.

1.1 Research Objectives

This work aims to investigate and explore the consequences of understandability metrics in regard to influencing the cognitive processes of software developers. To this end, we evaluate how the presentation of different values for an understandability metric impacts developers' expectations about understanding a code snippet. In the next step, we examine the relationship between expectations of comprehending code and motivation to engage with that code. Furthermore, we analyze the effects different expectations have on affective states. Additionally, we explore how differences between expectations about understanding a code snippet and the perception after inspecting the corresponding code snippet influence motivation and affective states. Lastly, we further investigate the cognitive bias that displaying an understandability metric has on the subjective comprehensibility assessment of programmers.

These objectives are addressed through an experiment in the context of a typical software engineering task of understanding a code snippet. In order to analyze and explain the results, we incorporate established theories from psychology and sociology about the concepts of expectancy, motivation, and affect with previous research in the software engineering domain.

1.2 Methodological Approach and Contributions

We conduct an experiment through an online survey with 81 participants that consist of a mix of students and IT professionals. As task, participants have to judge the understandability of a presented code snippet. Before and after this task, they are asked to answer questions about their expectations, motivation, and affective states. Participants are randomly assigned to one of three treatment groups to analyze the effect of different suggested understandability values: a control group, a group with a low understandability value, and a group with a high understandability value. Furthermore, we employ two code snippets that have different difficulty levels to inspect variations of reported perception, motivation, and affect as a result of the code snippet participants see. As a result, the contributions of this work are as follows:

- Different presented understandability metric values have no impact on the expectations of developers about understanding small code snippets. Therefore, understandability metrics do not compromise programmers' cognitive process with regard to expectancy.
- Expectations about understanding code positively influence the motivation to engage with that code. Our findings show a significant strong linear relationship between the two which should encourage anyone assigning programming tasks to ensure appropriate success expectations in the assignee.
- Code comprehension expectations based on small code snippets have no significant impact on affective states.
- Differences between expectations and perceptions regarding the understandability of code have a large influence on the motivation and happiness of developers. For that reason, it is important to guarantee the fulfillment of expectations.
- Understandability metrics do not affect the subjective understandability assessment of code in case of implausible metric values in combination with experienced developers as well as equal comprehension expectations. As a result, understandability metrics do not induce a cognitive bias under these conditions.

1.3 Thesis Structure

This work is structured in the following way:

Chapter 2 – Background and Theoretical Foundations: This chapter provides background information on source code understandability and corresponding metrics. In addition, it presents the theoretical foundations of expectancy, motivation, and affective states.

Chapter 3 – Related Work: We describe previous work about cognitive biases in software engineering, the influence of expectations, the relationship between expectations and motivation, and the impact of affective states on software developers in this chapter.

Chapter 4 – Methodology: The research questions, experiment design, used materials, and survey procedure is outlined in this chapter. Furthermore, we define our hypotheses and analysis approach.

Chapter 5 – Results: In this chapter, we present the results of our experiment which includes testing our hypotheses.

Chapter 6 – Discussion: We examine the results and discuss limitations as well as implications of our research in this chapter.

Chapter 7 – Conclusion: This chapter summarizes the work and outlines directions for future research.

Chapter 2

Background and Theoretical Foundations

This chapter highlights context and background information about the topics at hand. First, we provide an overview of code understandability. This section discusses definitions of code understandability and reports on the challenges of finding an adequate measurement for code understandability. Afterwards, the anchoring effect is explained since it is relevant for letting developers judge the understandability of a code snippet after we already presented them an understandability value for it. Then, we illustrate the theoretical foundations of expectancy, motivation, and affect theory. We present relevant psychological theories of these three concepts and how they correlate as well as information on how to measure them in an experimental setting.

2.1 Source Code Understandability

Understandability has been an important factor to evaluate the maintainability of software for a long time. As an essential part of maintenance, refactoring efforts aim, among other things, to improve the understandability of the software [Fow18]. Here, Fowler believes, that "understandability is next to godliness" [Fow18, p. 251]. As a result, it found its way into several software quality models as a key factor describing a prerequisite for any maintenance task since developers need to understand the software they work with before they can maintain it [ASC; BBL76; CAS+16; TN14]. For that reason, understandability and maintainability are closely related. In such context, understandability is considered on the whole software product level, combining source code with other documentation. In the most recent standard quality model, which is described in ISO 25010 [ISO11], understandability is not a specified quality attribute of maintainability. Instead, the most closely related existing quality attributes analysability and modifiability only describe aspects that are impacted by understandability.

While understandability is an important factor on the software product level, in this work we focus on the level of individual code snippets and how developers are impacted by associated understandability metrics.

2.1.1 Definition

Source code understandability still lacks a precise, widely used definition that encapsulates all its aspects. As a result, most authors shy away from using a specific definition and instead describe understandability through its function and importance or through the metrics they use to measure it. For this work, we use the original definition by Boehm, Brown, and Lipow [BBL76, p. 605]:

Definition 2.1.1 (Understandability)
Code possesses the characteristic of understandability to the extent that its purpose is clear to the inspector.

Furthermore, comprehensibility and comprehension are used as synonyms for understandability and understanding in this work. These synonyms are used with the same meaning in the literature, describing the process of understanding code written by oneself or others [AWF18; RW97]. Closely related to understandability are readability and complexity which are often defined with the same purpose [CSLB19]. Specifically readability needs to be differentiated from understandability. Scalabrino et al. [SBV+19] emphasize that certain code can be highly readable, while still being hard to understand as a result of missing knowledge. In their original quality model Boehm, Brown, and Lipow [BBL76] describe legibility, the quality of being clear enough to read, as a prerequisite for understandability. In the same vein, Borstler and Paech express that "readability is required for comprehensibility, but readability does not necessarily imply comprehensibility" [BP16, p. 887] which makes it complicated to measure them independently. As such, they view readability as a property of the code, while understandability is a characteristic of the reader. Similarly, Posnett, Hindle, and Devanbu consider readability as a syntactic aspect of code and understandability as a semantic aspect of code, where "readability is a perceived barrier to understanding that the programmer feels the need to overcome before working with a body of code" [PHD11, p. 73]. Other authors do not differentiate between readability and understandability, which furthers the unclear definitions in this area of research. Buse and Weimer define readability "as a human judgment of how easy a text is to understand" [BW10, p. 1] conflating the two into one construct.

Another concept which is part of readability and related to understandability is clean code [Edw00]. Proposed by Robert C. Martin, clean code describes a set of rules and principles that are meant to create elegant, simple, and direct code that can be understood by someone other than the original author [Mar09]. As such, clean code is very similar to the notion of readability since clean code is supposed to be easy to read. Therefore, the concept of clean code follows the same relationship as readability in regards to understandability.

The second related concept, complexity, is often used as a proxy for understandability. Complexity is thought of as an attribute which makes code hard to understand [AWF18]. However, similar to readability, complexity does not imply understandability, where an uncomplex code snippet can still be hard to understand as a result of other factors.

2.1.2 Measuring Understandability

It is very difficult to measure source code understandability because of the multitude of factors that play a role in its assessment. Depending on the instruments that are used to measure comprehensibility, the results can be highly influenced by the aspects developers consider for understandability. While some researchers opt for more objective measurements like time used to understand code [AVZ15; Kol16], comprehension questions [KW13], or bug fixing tasks [FALK11] others rely on subjective understandability assessments from the participants [BW10; SBV+17]. This subjectivity includes personal taste as a result of the readability aspects of understandability, as well as prior knowledge of the developer regarding the source codes language constructs and application domain [SBV+17]. In the end, it is not clear which instrument or combination of instruments capture understandability the best.

Over the years, many different methods of measuring the understandability of source code have been tried. Siegmund [Sie16] describes three approaches that are used historically, namely think-aloud protocols, memorization, and comprehension tasks. Think-aloud protocols are used to observe the cognitive processes during understandability tasks. The participants in corresponding studies are asked to describe out loud what they are doing and why. In studies using memorization methods to measure comprehension, developers have to recall source code they saw before. Lastly, through comprehension tasks, programmers are asked to show their understanding of code by answering questions about it or filling out blanks in the code.

In a systematic literature review, Muñoz Barón [Muñ19] analyzed 57 papers measuring understandability. The tasks used to evaluate comprehension in these studies are visualized in Figure 2.1. Asking specific comprehension questions is used most often to investigate whether developers did understand the code. These questions compromise of asking about the output of the code or letting the programmers describe the functionality of the code. A different way to measure understandability is to let developers subjectively

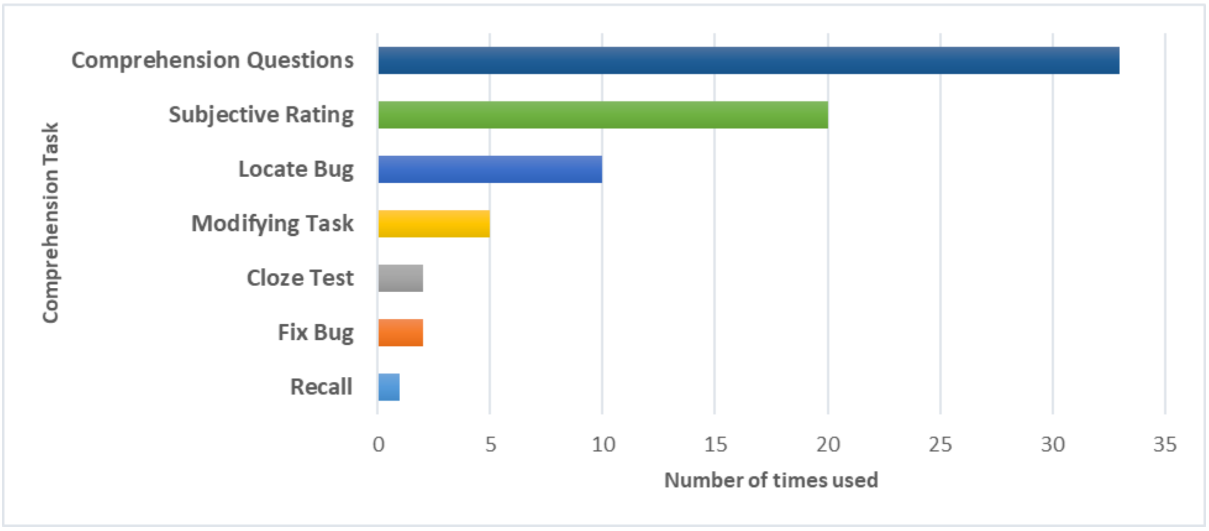


Figure 2.1: The different methods used to measure code understandability sorted by their occurrence in a systematic literature review [Muñ19].

rate the code on a scale from easy to understand to hard to understand, like a Likert scale [Lik32]. Other methods include tasks of locating and fixing a bug in the code. A small number of studies utilize more advanced modifying tasks like refactoring or extending parts of the code. Rarely, forms of cloze tests are used, in which participants need to fill out blank parts of a program. Lastly, asking developers to recall elements of a code snippet that they previously were shown for some time, is used in very few studies.

The ultimate goal of measuring understandability is to find a metric that can be operationalized. Such a metric could be used to automatically assess the understandability of source code. As a result, static analysis tools could show this metric to programmers in their development environments to indicate parts of the code that could be improved with a clear description of how to improve it [Sto05].

Software Metrics

The easiest way to operationalize an automatic understandability measurement would be through software metrics since they can be calculated fast and efficiently. While many metrics were tried to represent understandability, for almost all of them there currently exists no strong empirical evidence to actually measure understandability [CSLB19; SBV+17].

One prominent approach to measure understandability is through code complexity. The idea being, that more complex code is harder to understand. For this, McCabe's cyclomatic complexity (MCC) [McC76] is still, to this day, a widespread metric used for complexity. It measures the number of independent paths in the code and was originally developed as a testing-complexity metric [AWF18]. However, this metric received a lot of criticism. This includes the lack of theoretical foundation [She88], the nesting problem of not differentiating between loops of different iterations [SSA13], and that MCC does not reflect code complexity as it is experienced by humans [JF14]. Therefore, more recent approaches try to incorporate other aspects into complexity which lead to better metrics for understandability. Jbara and Feitelson [JF14] argue that code regularity, where you use the same structures repeatedly, leads to better comprehension while compensating for MCC and lines of code (LOC). Another method by Chhabra, Aggarwal, and Singh [CAS03] introduces spatial complexity of code and data, which is the distance between modules and variables to their different uses in the code, to better explain the cognitive effort needed to understand software.

Researchers have attempted to find many more metrics that represent code understandability with varying success. While analyzing student's exam results, the software metrics MCC, nested block depth, and two dynamic metrics based on executed statements were correlated with the difficulty of the code tracing tasks from the exam [KW13]. On the other side, Feigenspan et al. [FALK11] found no difference in program understanding while varying complexity, LOC, concern attributes, and concern operations between two software implementations. Additionally, clean code has been empirically shown to not immediately improve the understandability of the code [AVZ15; Kol16]. Ammerlaan, Veninga, and Zaidman [AVZ15] investigated legacy code in an industrial environment, to see whether refactored clean code would improve code understandability. They generated clean code versions of program code in three stages, namely small, medium, and large. Furthermore, they designed corresponding coding tasks and measure the time it takes to complete them. In two out of three of the small tasks and in the large task, the problems were solved faster in the group without refactored code. However, the developers in the group with clean code versions created better solutions, fixing the root cause of the problem. Similarly, Koller [Kol16] conducted an experiment with two groups solving three small coding tasks on either a legacy code or a refactored clean code. Two out of the three tasks are solved faster by the group with legacy code, which corroborates the result that clean code does not necessarily improve code understandability. Like Ammerlaan, Veninga, and Zaidman, Koller notes that developers working with clean code write higher quality code.

In an effort to explore correlations with understandability Scalabrino et al. [SBV+17] analyzed 121 different metrics. They considered three different types of metrics: code-related metrics, documentation-related metrics, and developer-related metrics. The 105 code-related metrics included typical metrics like MCC and LOC as well as readability metrics. They used 11 documentation-related metrics, which describe the availability of documentation for a given code snippet. Lastly, the five developer-related metrics assesses to what extend developer experience and background impact code comprehension. In a study with 46 participants trying to understand eight code snippets, they found no significant correlation between any of the metrics and the understandability of the code snippets. In a reanalysis of the data from Scalabrino et al., Trockman et al. [TCM+18] tried to capture understandability through a combination of multiple features. They used different statistical models and found a small but significant correlation between understandability and a combination of syntactic structure and documentation metrics. Additionally, they were able to create a classifier which can differentiate between hard to understand code and easy to understand code with small discriminating power. Trockman et al. suggest that it might be possible to find a useful understandability metric with more data. Following this attempt, Scalabrino et al. [SBV+19] reported a second analysis of their study with an increased sample size of 63 developers. The result remained the same, namely that there is no correlation between the 121 metrics and code understandability. Combining multiple metrics resulted in models with some discriminatory power and with higher correlation. However, Scalabrino et al. remark that these models are still far from being usable in practice.

One promising understandability metric, which was not part of the analysis by Scalabrino et al., is Cognitive Complexity. Cognitive Complexity was introduced in the SonarSource[1] environment by Campbell [Cam18a] as a measurement that reflects understandability more closely to programmers' intuition about the cognitive effort required to understand the code. Similar to MCC, Cognitive Complexity assesses code based on its structure, while overcoming many of the shortcomings of MCC. It ignores shorthanded versions of multiple statements, increments for every break in the linear code flow, and increments based on the nesting depth. Campbell [Cam18b] analyzed the acceptance of this new metric in 22 projects which used Cognitive Complexity through their SonarCloud[2] tool. She found an acceptance rate of 77% based on the projects that fixed problems associated with Cognitive Complexity. In a recent meta-analysis, Muñoz Barón, Wyrich, and Wagner [MWW20] validated Cognitive Complexity as the first code-based metric measuring understandability. They conducted a systematic literature search to find open data sets of understandability studies, resulting in a data set of 24,000 understandability evaluations from 427 code snippets. With this data Muñoz Barón, Wyrich, and Wagner calculated correlations between Cognitive Complexity and measurements of understandability. They found empirical support for a correlation with time, subjective ratings, and composite variables. While the results were mixed regarding correctness and physiological measures, they conclude that Cognitive Complexity is able to represent at least some aspects of code understandability.

[1]The SonarSource world: https://www.sonarsource.com includes plugins, software, and cloud resources to improve code quality in 27 different programming languages.
[2]https://www.sonarqube.org

Physiological Measures

A new approach to measure understandability are physiological instruments. Instead of focusing on the code, they focus on the human who has to understand the code. As a result, we can gain a better understanding of how code comprehension impacts the cognitive effort of developers. This can lead to improved tooling possibilities as well as recognizing understandability problems during the comprehension process instead of after they occur.

Siegmund et al. [SKA+14] used functional magnetic resonance imaging (fMRI), which measures the blood-oxygen levels that change as a result of brain activity, during code comprehension tasks. They found activation of areas relating to working memory, attention, and language comprehension. As one result of this gained knowledge, they theorize that increasing the number of variables beyond the capacity of the working memory should impair code comprehension [PSA+20]. Using wearable Near Infrared Spectroscopy (NIRS) devices, which measure cerebral blood flow, Nakagawa et al. [NKU+14] were able to show that they can identify developers working on hard code through higher blood flow measurements. Similarly, Fritz et al. [FBM+14] created a nominal code predictor (easy/difficult) which can predict whether a new participant will perceive their task as difficult with a precision of over 70%. To achieve this, they utilized an eye-tracker, an electrodermal activity sensor, and an electroencephalography sensor. Müller and Fritz [MF16] showed that biometrics outperform more traditional metrics, like code complexity, in predicting the difficulty a developer perceives while working on code. Additionally, they found that difficult code elements result in more quality concerns raised during peer code reviews. By using an eye-tracker and a minimally invasive brain imaging technique called functional Near-Infrared Spectroscopy (fNIRS), Fakhoury et al. [FMAA18] analyzed the effects of source code lexicon and readability on developers' cognitive load. Their results show that linguistic antipatterns significantly increase the cognitive load during program comprehension tasks. Additionally, a combination of structural and linguistic antipatterns leads to 60% of participants being unable to successfully complete the given task.

2.1.3 Cognitive Models

In order to investigate the human aspects of code understanding, we need to know how developers go about comprehending a piece of program code. Cognitive models try to explain how programmers understand given source code by step-wise building a mental model of the code they are inspecting. Understanding program code involves using existing knowledge of the programming language, programming principles, the programming environment as well as typical algorithms and solution approaches to gain new knowledge about the code [MV95]. Generally, there exist two approaches to build this mental model, bottom-up and top-down [Sie16]. In the top-down approach, programmers reconstruct knowledge about the software domain they know about and map it to the current source code [Bro83]. This process is guided through building a hierarchy of hypotheses that start with a top, high-level one which is supported by the subsidiary hypotheses. Using their domain knowledge developers confirm and refine their hypotheses while understanding the current source code. Especially experts follow this approach when the program follows a plan they know and expect [SE84].

Bottom-up comprehension considers developers creating a control flow abstraction first which includes the sequence of operations in the code they are trying to understand [Pen87]. In this process, they read individual program statements and chunk or group them together into higher-level abstractions [SM79]. After this so-called program model is fully built, developers create a situation model that contains data-flow and functional abstractions [Pen87]. With this approach, programmers build their mental model from individual statements up to a complete understanding of the code they are inspecting.

For this work, we consider the bottom-up comprehension model. We are using small, domain independent code snippets that can be understood without prior knowledge except for existing knowledge about the Java programming language.

2.2 Anchoring Effect

The anchoring effect as we understand it today was described by Tversky and Kahneman [TK74] and is defined as "the disproportionate influence on decision makers to make judgments that are biased toward an initially presented value" [FB11, p. 35]. It is one of the most pervasive and robust cognitive effects [FB11; MES04]. The anchoring effect has been shown to take place in many different areas. It occurs when the anchor values are clearly uninformative for the estimate like when they are generated by chance. Furthermore, it appears to be independent of participant motivation including monetary rewards for better estimations. In addition, the expertise of the participant plays at most a mitigating role in the effect. This means, even experienced judges in a particular field are influenced by an anchor. In the same vein, people with higher cognitive abilities are not as susceptible to the anchoring effect, but it is still sizable and significant. The effect can also persist over fairly long periods of time of up to one week. Strikingly, the anchoring effect influences participants even when they were warned beforehand about its potential distortion. In regard to implausible or extreme anchors, there exist mixed research results about their influence on the anchoring effect [FB11]. Some results suggest that extreme anchors create bigger anchoring effects, while others show the opposite. Furthermore, emotions as well as personality traits can affect the anchoring effect. For one, being in a sad mood results in being more susceptible to the anchoring bias. Additionally, people with different personality traits including high agreeableness and low extraversion as well as openness to experience are more vulnerable in regards to the anchoring effect.

This effect can be shown through different experiments where the anchor is provided by the experimenter, self-generated, or provided in an unrelated task [MES04]. The classical anchor experiment set-up consists of first asking the participants a comparative question in regards to the anchor and then an absolute anchoring question. Tversky and Kahneman [TK74] asked participants to estimate the percentage of African countries in the UN. The anchor was created through spinning a wheel of fortune between 0 and 100 in the participants' presence. Afterwards, they first asked the subjects to guess whether or not the percentage of African countries in the UN is higher than the anchor, which is the comparative judgment. Then the participants had to estimate the absolute percentage. The authors found a clear difference between groups, where the group with an anchor of 10 estimated a percentage of 25 on average, and a group with an anchor of 65 estimated a percentage of 45 on average.

Explanations of the anchoring effect in the literature differ [FB11]. Tversky and Kahneman [TK74] proposed that people make insufficient adjustments upwards or downwards based on the anchor values. Therefore, different starting points yield different estimates. However, this view does not incorporate anchoring effects based on plausible numbers, because there would be no need to adjust from them. Instead, Mussweiler, Englich, and Strack [MES04] suggest a model of selective accessibility. This model postulates that comparing the judgment to the given anchor value changes the accessibility of knowledge about the target. Specifically, anchor-consistent information is selectively more accessible. The final judgment is then heavily influenced by the accessible knowledge. A second explanatory theory is confirmatory hypothesis testing [FB11]. Similarly, this theory suggests, that the anchoring effect activates anchor-consistent information. However, it proposes that the subjects consider the anchor value as the correct answer and test out this hypothesis. Furnham and Boo [FB11] conclude that both, selective accessibility and confirmatory search contribute to the mechanism of the anchoring effect, while they see the confirmatory search approach as the dominant view.

2.3 Expectancy

Expectations are an integral part of peoples everyday life guiding their behavior [ORZ96]. Acting for the most part as unconsciously held background assumptions they serve humans as heuristics providing input into their judgments. As a result, individuals try to validate their held expectations. This can influence people's experiences since they want to see what they expect to see.

Expectations are built experience by experience [ORZ96]. Initially, they are narrow and specific, but as people accumulate experiences they extract generalizations that summarize events across stimuli, time, and situations. As a result, these expectations become more general and broader before they finally settle in the middle balancing breadth and death of specific events to general assumptions. Afterwards, peoples expectancies guide their behavior in a self-perpetuating manner. If an individual found expectancies useful in the past then their cognitive system is rather cautious about altering or replacing them. Specialized processes exist to handle unexpected events and integrate them into their model to improve expectancies. In the end, these expectancies are subservient to humans primary goal of effective behavior, which means that they are changed when more useful information is experienced [ORZ96].

When people make predictions, like how well they will understand a code snippet, they tend to bring information to mind that is consistent with their prediction [ORZ96]. This can result in overconfidence. At the same time, believing in one's own future success facilitates this success. Expecting to succeed affects behavior in a way of increasing confidence [Fea66] and task persistence [CBS79]. Additionally, specifically thinking about such expectations fosters intentions that guide ongoing behavior in regards to the underlying matter [ORZ96]. Furthermore, optimistic expectations produce positive affect, which is generally motivating and energizing.

While evaluating new events in correspondence to their expectations people aim for processing fluency [ORZ96]. This concept describes the extent to which pattern matching flows smoothly in contrast to being interrupted by expectancy disconfirmations. Here, processing fluency is the first point where the cognitive system registers a confirmed or disconfirmed expectancy resulting in related cognitive consequences. During this continuing evaluation of expectations to perception, affective responses act as signals regarding goal progress. Positive affect signals sufficient progress while negative affect signals insufficient progress. This progress is evaluated through a small or large discrepancy between expectation and current status. Therefore, when an expectancy is disconfirmed people tend to have a response of negative affect. Any perceived bad outcome results in negative affect, but when it is unexpected it is even more extremely dissatisfying. In the same vein, positive outcomes feel all the sweeter when it is a surprise to the perceiver [ORZ96].

These findings were adapted into their own theory called expectancy-confirmation theory [Bha01] which finds most of its use in marketing. This theory posits that individuals first develop an expectation with respect to a product or service or software artifact. After using the specified product a perceived performance is established. Then, the expectation is compared to the perceived performance which either confirms or disconfirms the expectation. As a result, the level of disconfirmation determines the individual's satisfaction and therefore influences the affective state [Bha01].

Since expectations are an essential part of humans, they are also an important factor in many psychological effects [SPB16]. These effects can have a significant impact on behavior and are able to create a lasting influence on cognitive operations. One relevant effect is explained in the subsection below.

2.3.1 Placebo Effect

"A placebo effect is a genuine psychological or physiological effect, in a human or another animal, which is attributable to receiving a substance or undergoing a procedure, but is not due to the inherent powers of that substance or procedure" [SP04, p. 326]. The placebo, in this instance, is the corresponding substance or procedure which elicits the placebo effect. Many different types of placebo responses have been identified, that are driven by varying mechanisms depending on the context [PFB08]. The placebo effect is mostly known from clinical research into drugs, pain, depression, anxiety, and other treatments. However, nowadays it is recognized, that the placebo effect can be shown in plenty of other contexts. Researches have demonstrated, that the placebo effect can enhance sports performance [HSS+19], cognitive performance [TBG+18] as well as creativity [RMI+17]. Therefore, at its core, the placebo effect explains how the context of beliefs and values induce brain processes that have an effect on perception and emotion resulting in mental and physical outcomes [Ben05].

One prominent explanation of the placebo effect is expectancy theory [PFB08]. According to this theory, the "placebo produces an effect because the recipient expects it to" [SP04, p. 328]. Therefore, the placebo effect is a subcategory of expectancy effects. This does not exclude other co-founding factors like the relationship with the placebo administrator, or sociocultural aspects. However, they are all influencing through the expectations of the recipient. There exists a large body of literature investigating the profound impact of expectations in the placebo environment [SPB16]. One simple example is the difference between open and hidden administration of medication. When a patient is unaware of a treatment, it is less effective and takes longer in contrast to the treatment being openly communicated [CLLB04]. Here, the expectations raised in the patient for the treatment outcome and through the patient-doctor relationship play an integral role in this difference. Furthermore, in pain relief studies it was confirmed, that expectancy but not the desire for relief was responsible for a difference in the placebo pain relief [PMK+99]. Additionally, further research into placebo pain mechanisms revealed, that expectations are not influenced by personality traits and can predict the placebo response independently [CC17]. Nowadays our understanding of the placebo

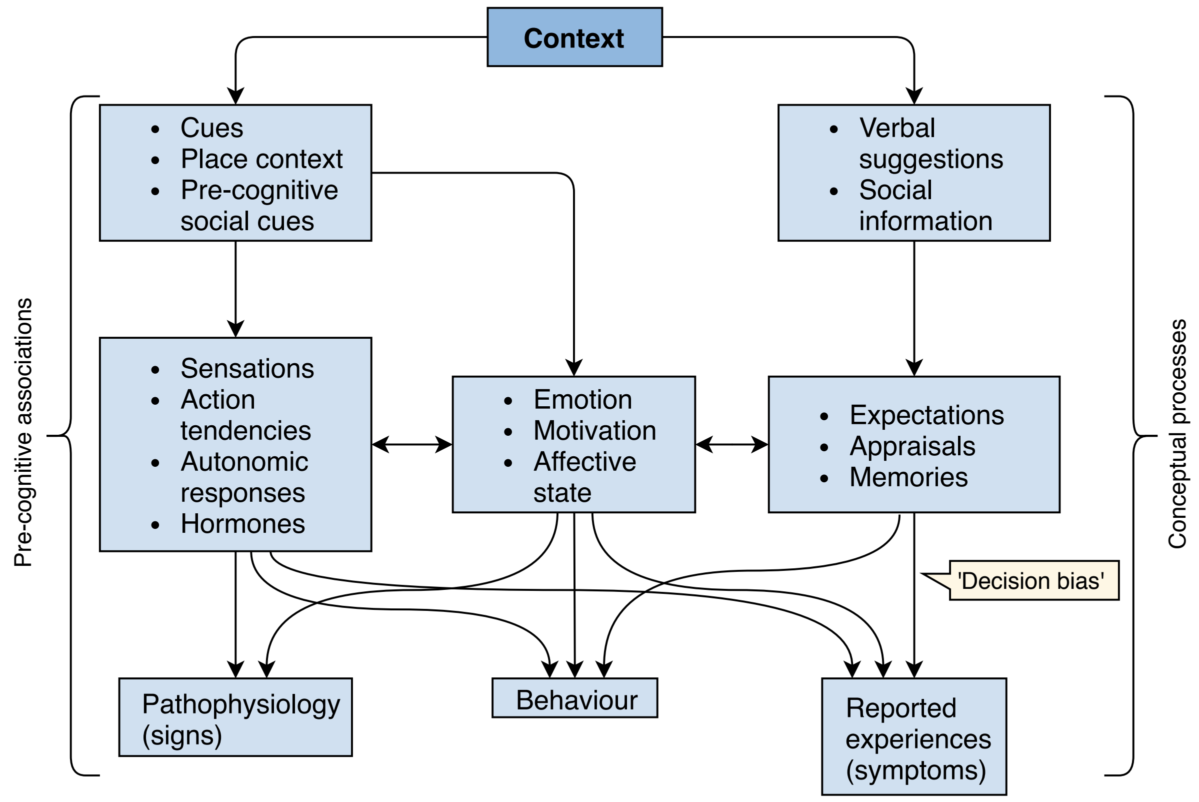


Figure 2.2: Framework illustrating the context and the processes involved in the placebo effect [WA15].

effect is, that some placebo mechanisms work mostly through expectancy, some mostly through classical conditioning, and others through a combination of both [PFB08; SP04; WA15].

Wager and Atlas [WA15] conclude that all placebo effects are created in a context of social and physical cues as well as verbal suggestions and history. This context integrates diverse psychological elements, including learned associations, past experiences, and expectations elicited through verbal suggestions and social interactions. Using this information Wager and Atlas present a framework of three types of psychological factors that induce placebo effects, namely pre-cognitive associations, conceptual processes, and affective or motivational states. This framework is shown in Figure 2.2. Pre-cognitive associations are independent of what a person believes or expects and elicited through conditioned cues. They influence physiological processes outside conscious control, which can impact emotion, motivation, and affective states. On the other side, conceptual processes are dependent on thoughts, expectations, and memories. They are evoked by verbal suggestions and background beliefs and influence emotional and motivational states. In the end, these processes differentially affect outcomes of reported experiences, behaviour, and physiology.

2.3.2 Measuring Expectations

Expectations are typically measured by asking the subjects about them before any kind of treatment. Oftentimes this is done relatively informally through questions where the participants have to select their outcome expectations on a simple scale [BSSS13; CC17; PMK+99].

However, there exist many empirically validated expectancy scales for different contexts. Most of them are centered around the topics of medical treatments or education. For medical treatments, there are whole purpose scales that ask about beliefs and fears regarding upcoming treatments with multiple items [DB00; YGHM12]. Furthermore, scales for specific therapies exist like acupuncture [MAFB07; MXB10]. On the other side, scales regarding education focus on students expectations concerning their success in school classes or topics [KHBG14], test-taking in general [STB00], or education training activities [ZFT+11]. In these last three instances expectancy is used as a proxy for motivation, based on different motivation theories, which are further discussed in Section 2.4.

These scales are validated through oftentimes multiple studies with hundreds of participants. They focus on internal reliability, validity, and retest accuracy. One factor, such as expectancy, is measured through multiple items where the individual scores are added up to one factor score. Each item consists of a statement, which is measured by subjects indicating their agreeableness on a Likert scale. The goal is to create a scale, that is easily and fast administrable, while still having enough items to be internally reliable.

2.4 Motivation

"Motivation is the psychological processes that cause the arousal, direction, and persistence of behavior" [Mit82, p. 81]. It is an individual as well as intentional process that guides behavior. Here, motivation theories are concerned with internal and external factors that influence the choice of action of individuals. A manifold of different theories exist that are applied depending on the context and research domain. While some researchers attempt to create integrated [SK06] or general [Bau15] motivation theories, currently the individual ones are used in practice.

Independent of other motivation factors, researchers found that setting specific and challenging goals linked to feedback on results leads to the most effective performance [Lun11b]. This goal-setting theory of motivation [LL02] postulates that goals represent something humans are consciously trying to do, which creates a desire to get involved in corresponding actions. Furthermore, having a challenging goal leads to higher effort and persistence. As a result, reaching set goal creates satisfaction and further motivation while the opposite causes frustration and lower motivation. Additionally, a learning goal orientation, where a person wants to acquire new knowledge by mastering challenging situations, leads to a higher performance than a performance goal orientation in which one demonstrates their abilities [Lun11b].

Another relevant group of theories are expectancy theories of motivation. Many different versions of this correlation have been developed over the years [BH14]. One early attempt by Vroom [Vro64] describes motivation as the combination of valence, instrumentality, and expectancy (VIE). Here, valence is defined as the attractiveness of an outcome. Instrumentality describes the belief that putting in some degree of effort into the activity will lead to the desired outcome. Lastly, expectancy expresses the subjective probability that this effort will result in the outcome. Vroom suggests that these three factors are in a multiplied dependence to motivation [Lun11a]. This means the highest motivation is achieved when all three are high, while the motivation can be zero if one of the three is perceived to be zero. A more recent theory of motivation is the expectancy-value theory by Eccles et al. [EAF+83]. This model proposes that motivation consists of two key factors: expectancy and value. It postulates that to be fully motivated one has to believe they can do a task and see value to want to do the task. Therefore, expectancy in this context describes the success belief of a person in regards to a goal. It is thought of in two dimensions, the belief in ones current and future ability to succeed. However, in studies, they found these dimensions to be indistinguishable and therefore suggest to handle expectancy as one individual factor [EW95]. The second factor, value, reflects to what extent a person thinks some task is beneficial. Its dimensions distinguish between positive and negative influences. Positive influences are intrinsic value (inherently enjoyable), utility value (helping in the achievement of goals), and attainment value (affirms valued aspect of a person's identity). On the other side, potential costs in regards to alternative activities, effort, and time as well as negative psychological states are considered negative influences. Therefore, Barron and Hulleman [BH14] argue that cost should be its one category with the described sub-components resulting in an expectancy-value-cost model.

Furthermore, self-determination theory (SDT) has been applied in a lot of motivation research. Introduced by Deci [Dec86] SDT describes varied intrinsic and extrinsic factors of motivation. It is built upon the belief that to foster motivation three innate psychological needs have to be satisfied, namely competence, autonomy, and relatedness. Competence relates to the need to be effective in one's environment [GVB00]. Secondly, autonomy describes freedom of pressure and the ability to choose one's actions. Lastly, relatedness describes interpersonal attachments between individuals referencing the fundamental strive of contact with others. When these needs are met intrinsic motivation, which is the strive to seek out activities that elicit pleasure and satisfaction inherent in the activity, flourishes [Dec86]. In contrast, extrinsic motivation describes various aspects that influence behavior beyond those inherent in the activity like a feeling of obligation or seeing it as a means to an end. SDT postulates that extrinsic motivation exists on a self-determination continuum from high to low external regulation [RD00]. On one extreme of this continuum is amotivation, a state of lacking any intention to act. Continuing, the next is extrinsic motivation with four sub-categories: external regulation, introjection, identification, and integration. They describe to what extent external factors are congruent with one's internal value system. Lastly, on the other extreme is intrinsic motivation, which is not regulated by external influences. Research shows, that people with more intrinsic motivation have more interest, excitement, and confidence resulting in enhanced performance, persistence, creativity, self-esteem, and general well-being [RD00].

2.4.1 Measuring Motivation

Motivation is measured in different ways, including observable cognitive, affective, behavioral, and physiological responses as well as through self-reports [TF14]. Additionally, motivation can be measured in a relative sense, compared to previous levels of motivation or compared to a different context. One way to measure motivation is through the degree of goal-related concepts that are accessible in memory. Since goals are an important part of targeted motivation, they activate goal-relevant information. As a result, this information is easier recognized, noticed, and remembered. This extents to the idea of measuring people's evaluation of objects that either further or hinder the goal, which evaluates goal congruent behavior. Another prominent approach involves measuring persistence through perceived choice. This consists of an experimenter leaving participants alone for some time and observing, whether the participants continue to work on the experimental task or do something else. Additionally, performance measures are used for motivation including accuracy, amount, highest level of achievement, and speed. However, especially the last one can have multiple interpretations. Slow speed could mean that the person has low motivation to engage with a task, or that they might savor the task because they like it so much, or that it is very important for them to do it right, or they might be just tired. With these measures, it is important to keep non-motivational factors constant across conditions, since as experimenters we do not know the exact cause of possible motivation depletion, whether it is because of motivational or physiological reasons [TF14].

In a lot of cases, researchers rely on self-reports to measure motivation. These are adminstrated through validated motivation scales based on the theories discussed before. Many scales exist covering different areas of motivation. Most notably, researchers in the area of education try to analyze ways to motivate students. Here, scales exist for general student motivation [Mar01], towards specific topics like science [TCS05], or physical education [DSC13]. Furthermore, achievement motivation [LF06] and task motivation [MCF01], as well as situational motivation [DVMK17], are evaluated in regard to how they impact students learning and success. However, motivation is assessed in many more areas like participation in fantasy football [DK11].

Additionally, many scales are based specifically on self-determination theory. In accordance with this theory, the intrinsic motivation inventory (IMI) [DR20] has been developed over multiple years and iterations. This scale analyzes intrinsic motivation through seven sub-scales: interest/enjoyment, perceived competence, effort, value/usefulness, felt pressure and tension, perceived choice, and relatedness. However, relatedness only applies in situations with multiple people and is therefore the least validated sub-scale. The interest/enjoyment subscale is the only one that assesses intrinsic motivation per se, while the others assess predictors of intrinsic motivation. The psychometric properties of this scale are validated in many studies and contexts [LWBM10; MDT89; MMP15]. Furthermore, scales based on SDT and specifically the IMI are adapted to be used for situational motivation [GVB00], academic motivation [FHFB05; VPB+92], leisure activity motivation [WB95], sport motivation [MKN+07], and motivation in spatial training through virtual reality [CS17].

As with expectancy scales, motivation scales are validated through multiple studies with hundreds of participants focusing on construct and discriminant validity, internal consistency, and temporal stability. Similarly, they consist of statements that are evaluated by participants through Likert scales. Each factor of a specific motivation scale is usually assessed through multiple items in order to improve internal consistency. As a result, motivation scales almost always consist of more than 10, oftentimes over 20 items.

2.5 Affective States

Humans encounter the world through emotions and moods. In every moment emotions play a part, mostly unconsciously, but sometimes they become very present as a result of an event. Everyone has an innate understanding of particular emotion concepts like fear, anger, sadness, and how they feel as well as what they entail [Rus03]. These innate concepts are partly inherited and partly learned through other humans and culture. However, these concepts vary depending on the context. Being afraid of an actual bear in the wilderness is quite different from feeling fear watching a horror movie or being afraid to miss the airplane. Furthermore, specific emotions can differ substantially in other cultures and languages [Rus91]. As such it is a challenging task for scientific researchers to define emotions in a way that encompasses all these possibilities. As a result, there does not exist an agreed upon definition of emotion and correlating concepts like mood, affect, and feeling [KK81; Rus03]. Most researchers agree, that emotion is a multifaceted phenomenon, but they differ on the specific components that are important and necessary [KK81; Moo09]. These components include a cognitive one, a feeling one, an emotional experience, a motivational one, action tendencies or action readiness (e.g. to flee), a somatic component, and a motor component (e.g. facial and vocal expression in accordance to fleeing) [Moo09].

Theories of emotion can be generally divided into two frameworks, the discrete and the dimensional framework [GWA15b]. In the discrete framework, theorists believe that a certain collection of basic emotions exist, which can be uniquely identified. Izard [Iza13] proposes the differential emotions theory, which is based on ten (interest, joy, surprise, distress, anger, disgust, contempt, fear, shame, and guilt) basic emotions. They are assumed to have innate neural substrates, a unique and universally recognized facial expression, behavioral consequences, and a unique feeling. Similarly, Ekman [Ekm92] beliefs that each emotion has unique aspects of signal, physiology, and antecedent events and that a set of basic emotions can be evaluated based on nine characteristics. Furthermore, Plutchik [Plu82] proposes a model of eight bipolar basic emotions, namely: joy and sadness, anger and fear, trust and disgust, surprise and anticipation. Mixing these basic emotions creates all other possible, so-called secondary emotions. Additionally, emotions can vary in intensity and persistence. According to Plutchik, the primary emotions are selected and based on basic adaptive behavior patterns that serve our survival instincts.

In the dimensional framework, researchers believe that emotions can be categorized across major dimensions. One such distinction is between the dimensions of positive and negative affect [WT85]. Watson and Tellegen [WT85] found positive and negative affect to consistently emerge as the primary and relatively independent dimensions in many affect studies and in major lines of mood research. They suggest, that these dimensions can be differentiated based on emotion terms serving as pure markers of either positive affect or negative affect [WCT88]. A different approach to emotion dimensions is based on the Pleasure-Arousal-Dominance (PAD) model. Here, studies indicate, that emotional states can best be described along the three independent and bipolar dimensions of pleasure-displeasure, degree of arousal, and dominance-submissiveness [RM77]. The pleasure dimension (also called valence) describes the range of pleasant to unpleasant emotion [LCD99]. Arousal is defined as the intensity of emotional activation. Lastly, dominance refers to feeling between total lack of control to feeling influential and in control [RM77]. In this theory, a person is viewed to be in an emotional state at every moment in the three-dimensional space of PAD. Additionally, bipolar means that each dimension is a continuum where a person can not experience high arousal and low arousal at the same time. The pleasure and arousal dimensions are seen as primary, while the third, dominance, is sometimes omitted [LCD99]. As such, any affect word can be defined as a combination of the pleasure and arousal component [Rus80].

In order to create a unifying theory of emotion, combining both frameworks as well as other theories, Russell proposes the concept of core affect [Rus03; Rus09]. Core affect is defined as "a neurophysiological state that is consciously accessible as a simple, nonreflective feeling that is an integral blend of hedonic (pleasure–displeasure) and arousal (sleepy–activated) values" [Rus03, p. 147]. Therefore, every person is in a state of core affect at any time, which is a combination of a pleasure and arousal value. Core affect is thought of as the most atomic unit which can be felt without any obvious stimulus, label, or attribution. Here, the feeling is an assessment of one's current condition. Other concepts, like emotion and mood, are then built upon this idea. Accordingly, mood is defined as a prolonged core affect with no object. Changes in core affect result from combinations of events. Sometimes, these changes can be attributed rather obviously, but at other times a change can be undergone without knowledge about why. A person attempts attributions and interpretations of their core affect all the time. However, since people do not have direct access to these causal connections, misattributions happen. When a causal link between events is perceived to be responsible for a change in core affect it is called attributed affect. It is defined by three necessary features: a change in core affect, an object, and an attribution of the core affect to the object. Attributed affects happen commonly in everyday life, like being happy about a good grade, or being afraid of a wasp. This concept is oftentimes the start of an emotional episode. In Russel's theory, emotional episodes such as fear and anger compromise a set of complex inter-correlated components that happen in prototypical cases. However, he recognizes that less prototypical cases are more common in everyday life. In essence, Russel suggests, that "emotional life consists of the continuous fluctuations in core affect, in pervasive perception of affective qualities, and in the frequent attribution of core affect to a single Object, all interacting with perceptual, cognitive, and behavior processes" [Rus03, p. 152]. As a result, the theory of core affect unifies previous emotion theories, while maintaining compatibility with the majority of the existing measurement instruments, regardless of them being about moods or emotions [GWA15b].

In the end, core affect guides cognitive processing and therefore influences humans memory [Rus03]. Furthermore, it impacts behavior from reflexes to complex decision making. Thereby, affect significantly shapes how people see the world, how they view themselves, how they remember past experiences, and what they do.

Therefore, we follow Graziotin, Wang, and Abrahamsson's suggestion for psychoempirical software engineering to use Russell's concept of affective states (affects) as our underlying theory of emotion [GWA15b]. As such, we understand affect to be the fundamental concept of the state of mind of developers. Accordingly, emotions can be seen as affect raised by a stimulus [Gra16].

2.5.1 Measuring Affective States

Following the various aspects of affects, measurements differ in their application between analyzing experiential, physiological, and behavioral responses [MR09]. Researchers suggest to use multiple assessment strategies, if possible, in order to understand affective states [KDL13; MR09]. Oftentimes its is complicated to evaluate to what degree a particular indicator is reflective of an emotion [KDL13].

However, most often self-report measurements of affects are used [KDL13]. One notable instrument is the Positive and Negative Affect Schedule (PANAS) [WCT88]. PANAS assesses the dimensions of positive and negative affect each with 10 items of corresponding affect terms. Therefore, it uses discrete emotions to measure two dimensions. However, it has been criticized to omit important emotions like joy while including terms that are not considered emotions like strong and active [DWT+09b]. Furthermore, several redundant items have been identified [Tho07]. As a result, more recent scales of positive and negative affect have been proposed, with fewer but meaningful items. Diener et al. [DWT+09b] introduce the Scale of Positive and Negative Experience (SPANE) [DWT+09a]. This scale evaluates the two dimensions through six items each. Participants are asked to indicate the frequency they felt a specified affect in the last four weeks which serves as a useful assessment of pre-existing affect. In the original study as well as follow-up research it has shown good psychometric properties [DWT+09b; LBW13].

In accordance to the PAD model of emotions, a pictorial assessment method has been developed to measure valence (pleasure), arousal, and dominance of a person's affective reaction to an object or stimulus [BL94]. The Self-Assessment Manikin (SAM) [BL94; LBC+97] consists of a set of figures for each dimension visualizing the potential affective states a person could feel. SAM is shown in Figure 2.3. The first row represents the valence dimension, which ranges from a frown to a smile. On the second row, the figures range from a calm, peaceful face to an exciting, explosive face indicating the difference in arousal. Lastly, the third row showcases the range of a small, submissive figure to a big, ubiquitous figure. This represents the dominance dimension. Each row functions as a 5-point scale for participants to indicate their current affective state. SAM eliminates the problems of finding the best affect terms as well as possible cross-cultural problems while still being quick and easy to use [BF17]. Its psychometric properties have been evaluated across a variety of settings demonstrating a reliable measurement [BF17; BL94; MWGK02]. Furthermore, SAM is consistently developed to be applicable

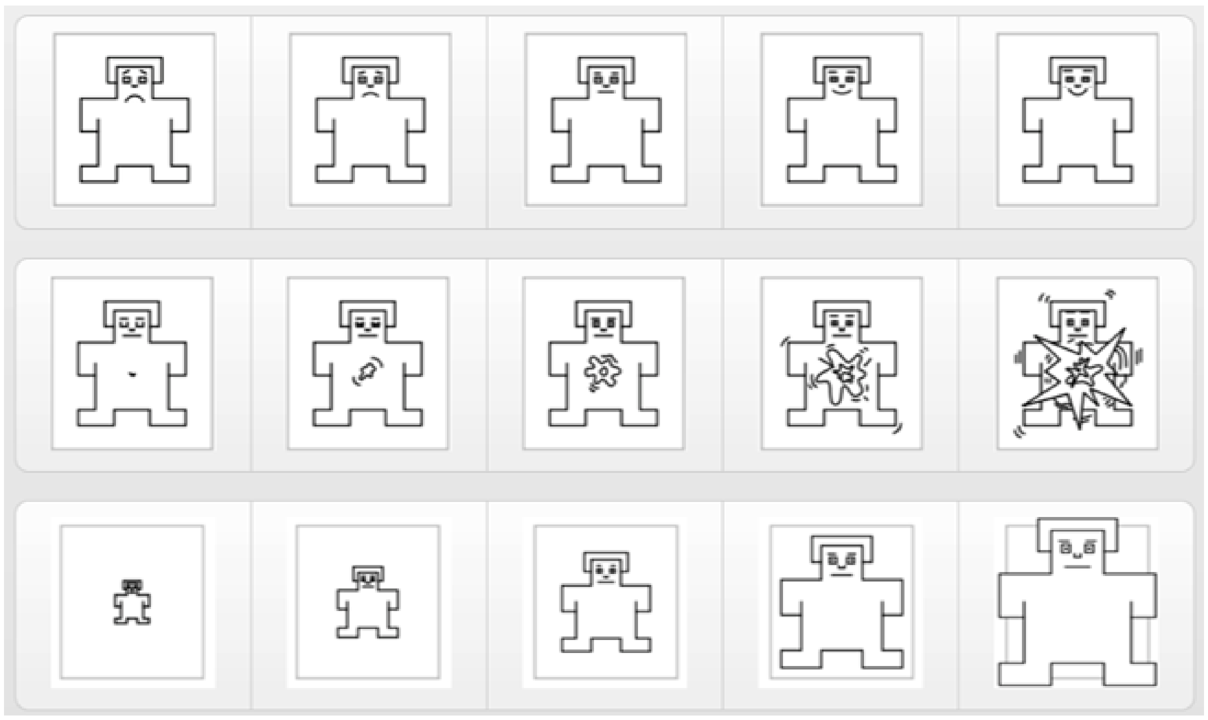


Figure 2.3: The Self-Assessment Manikin, which is used to measure valence (top-row), arousal (middle-row), and dominance (bottom-row) as affect dimensions [GWA15b].

in more contexts, like modernizing it into an affective slider [BV16], transforming it into emoticons for children [HPMB16], or into a tactile version for blind or visually impaired people [IM19].

Chapter 3

Related Work

In this chapter we present previous research, to place our work in the context of existing literature, and to compare our results with their findings. First, we review work regarding cognitive biases in the domain of software engineering. Afterwards, we highlight the problems of static analysis tools and the metrics that are used in them which was one factor that led to the current work. Then, we discuss the importance of measuring expectations and their influence on motivation. Lastly, we report on influential progress that has been made investigating the impact of affective states on developers' performance.

3.1 Cognitive Effects in Software Engineering

Our current work builds directly upon previous research by Preikschat [Pre20]. His goal was to understand how a manipulated understandability metric influences the performance of programmers. The aim was to investigate whether such a manipulated metric can have a placebo-like effect in developers working with source code. Therefore, he conducted a controlled, double-blind experiment with 45 participants split into two groups of developers that each had to solve the same three programming tasks. One group was told and shown during the task, that a new validated understandability metric based on machine learning assesses the code snippets to be easy to understand (4 on a scale from 0 to 10), while the other group was told they are hard to understand (8 on the same scale). The tasks involved calculating the correct output of the code snippets based on the code as well as based on the documentation since some errors were induced in the code. Furthermore, the time to complete the tasks was measured. Additionally, each participant subjectively judged the understandability of the three code snippets on the same scale that was used by the machine learning method. Preikschat analyzed whether this subjective judgment was influenced by the previous shown easy to understand or hard to understand metric and found a significant ($p < 0.001$) and large effect (Cohen's d = 1.376). He attributes this result to the anchoring effect, which we reviewed in Section 2.2, and proposes to consider this effect in any further experiments where metrics are involved. Furthermore, an analysis of performance in terms of correctness and time used between the groups showed no significant difference. Therefore, the experiment did not demonstrate a placebo effect in terms of performance. However, the author beliefs, that a different calculation for code understanding or a different manipulation could produce other results. In another step, Preikschat evaluated personality traits and whether they influence the previous results. In order to evaluate these aspects, he used SPANE for negative and positive affect, the Life-Orientation-Test for optimism and pessimism, and the Big Five personality test. The author discovered that lower anxiety and negative affect result in more deviation from the presented metric value. Theses results are in contrast to other literature that found different personality traits to be important. Therefore, Preikschat suggests to further explore the topic of personality traits in the software engineering context.

In a systematic mapping study Mohanani et al. [MST+18] analyzed the literature on cognitive biases in software engineering. Cognitive biases are defined as "systematic deviations from optimal reasoning" [MST+18, p. 1]. They reviewed 65 articles that investigate 37 cognitive biases organized into eight categories. The most investigated categories are interest (e.g. confirmation bias) and stability (e.g. anchoring bias) biases while social and decision biases are investigated the least. As a result, individual biases that are analyzed the most are the anchoring effect (26), confirmation bias (23), and overconfidence bias (16). However, the authors used the broadened definition of anchoring effect that includes the fixation on any kind of initial information, not only numbers. For software engineering, the anchoring effect is prominently found in estimation efforts of time or cost. Furthermore, the authors investigated to what extent possible debiasing techniques are proposed and evaluated. They found techniques only for 6 out of the 37 cognitive biases while none provided strong empirical evidence for their effectiveness. As a result, Mohanani et al. suggest to improve the research in four ways: conducting more qualitative and multimethodological research, investigating neglected areas, better integrating results across studies, and addressing confusion [MST+18, p. 13]. Additionally, they conclude that fundamental psychological and sociological mechanisms that affect these cognitive biases are poorly understood.

Corroborating these results, Chattopadhyay et al. [CNA+20] conducted a field study with ten developers to analyze which cognitive biases appear in the real world and how they impact developers. They observed 28 cognitive biases out of the 37 which are reported by Mohanani et al. [MST+18]. Chattopadhyay et al. categorized these biases into ten categories, from which the fixation category, which includes anchoring and adjustment biases, appeared most often. Analyzing the actions of developers they found, that reversal actions, which are actions that need to be undone, redone, or discarded later, were significantly more likely to occur with a bias. This is evidenced by 70% of actions that end up being reversed being associated with at least one cognitive bias. Furthermore, these biased actions had negative outcomes in terms of lost developer time. Out of the reversal actions, fixation played a role in more than half of them. Following the consequences of the observed biases, Chattopadhyay et al. recognize four categories of consequences: inadequate exploration, reduced sense-making, context loss, and misplaced attention. In addition, through follow-up interviews with more developers, they revealed, that these developers lack tool support to identify and prevent these biases.

Our work aims to extend the research by Preikschat [Pre20] and explore more fundamental consequences of understandability metrics. First, we want to investigate whether we can reproduce the anchoring effect that Preikschat found in the subjective assessment of code understandability. Beyond that, we follow the suggestion of Mohanani et al. [MST+18] and analyze the underlying psychological and sociological mechanisms of expectation, motivation, and emotion to see what role they play in developers handling of code understandability metrics.

3.2 Static Analysis Tools and Metrics

Static analysis tools are regularly integrated in various software projects [VPP+19] and have become a staple in big tech companies [SAE+18] in order to help developers find problems, bugs, and other areas that might be worth improving. These tools use rules, heuristics, and software metrics to calculate and display their results. However, researchers have raised concerns about the validity of the used metrics [NAG19] and their representativeness of underlying software quality characteristics as they are understood by developers [PLB18].

Already in earlier days, Shepperd and Ince [SI94] investigated the validity of three popular metrics at the time, including McCabe's cyclomatic complexity [McC76]. They found multiple problems with the fundamental model behind as well as technical problems within all three metrics. Furthermore, the metrics are used beyond their original context and lack clear empirical validation. Still, at least two out of the three metrics are used as software metrics and in static analysis tools until today [SBV+17]. Shepperd and Ince critique the poor understanding and omission of sufficient models underlying the investigated metrics. In their eyes, many of the problems are a result of substandard foundation and methodology. Therefore, they conclude, that "metrics based on flawed models are worse than valueless: they are potentially misleading" [SI94, p. 206]. Wedyan, Alrmuny, and Bieman [WAB09] analyzed the repositories of two open source projects (OSS), extracted fault fixes and refactorings, and compared them to reports of three static analysis tools. Their results show, that the analysis tools detected less than 3% of faults but at least up to 71% of refactoring concerns. Therefore, the inspected tools proved to be not effective and developers need to examine many false positives in order to find anything substantive. Furthermore, Nilson, Antinyan, and Gren [NAG19] investigated whether the metrics used in current static analysis tools are validated empirically. They evaluated metrics present in the literature and found 30 empirically validated metrics. From these only metrics that correspond to external quality attributes were selected which results in a final list of 12 metrics. Furthermore, Nilson, Antinyan, and Gren analyzed 130 analysis tools from which only six satisfied their selection criteria. They found, that the validated metrics are poorly represented in the selected analysis tools with no tool supporting more than half of these metrics. Instead, the analysis tools provide an overwhelming amount of metrics ($\sim$96%) which are not empirically verified and therefore have an unclear purpose. The authors conclude, that this might confuse developers. Additionally, it raises a question about what all the other metrics are supposed to represent.

With a different approach, researchers have recently evaluated the validity of metrics by comparing commits in software projects to changes in metrics that are supposed to measure the intention of the commit [AMOK19; FRHA19; PLB18]. The idea involves mining commits in open software projects that mention certain quality aspects and analyze whether the mentioned aspect is measurable through prominent corresponding metrics from the literature. Through this method, researchers want to investigate whether metrics are able to represent software qualities as they are perceived by developers. The first one to apply this approach was Pantiuchina, Lanza, and Bavota [PLB18], who analyzed qualities regarding refactorings, namely cohesion, coupling, code readability, and code complexity. They mined 1282 commits in which a clear intention to improve one of the four qualities was made and compared it to multiple proposed metrics for the corresponding quality. They found, that in most cases the metrics designed to reflect certain quality aspects were not able to capture quality improvements as seen by developers. Following this attempt, AlOmar et al. [AMOK19] investigated eight quality attributes with the same method. They classified 1245 commits according to the claimed quality improvement and evaluated 27 metrics that are assumed to measure these effects. The authors identified metrics for five out of the eight quality attributes that can capture developers' intentions of quality improvement. Fakhoury et al. [FRHA19] specifically inspected code readability improvements. Comparing 548 commits of readability improvements to three state of the art readability models, they found that all three models fail to capture readability improvements. However, in examining additional metrics they identify candidates that are successful in detecting readability improvements, which should be considered for future models.

On the other side, research has continuously investigated how developers view and interact with static analysis tools. Consistently the biggest problem of users is the high number of false positive results and the difficulty in understanding the warnings of static analysis tools [IRFW19; JSMB13]. To this end, Johnson et al. [JSMB13] interviewed 20 experienced developers about their use of static analysis tools. Most unsatisfactory for them were the high amount of warnings and uninformative presentation of these warnings leaving them with questions about what the problem is and how to fix it. Coming to the same results, Imtiaz et al. [IRFW19] evaluated Stack Overflow questions regarding static analysis tools. Most questions relate to how to ignore or filter alerts (23.9%), asking for validation of false positives (22.9%), and how to actually fix a certain warning (19.6%).

We believe that these problems developers have with static analysis tools are in part a result of the above outlined widespread use of unvalidated metrics in analysis tools. Combining these aspects shows a concerning view of static analysis tools and the metrics used to facilitate their results. Furthermore, it indicates potential problems for developers who use them. As such, in this work, we want to provide a different perspective on the relationship between metrics and developers. This is why we investigate to what extent these possibly inaccurate metrics have an effect on expectations, motivation, and affective states. Since these aspects affect performance, persistence, and general well-being it is important to understand their interplay in this context.

3.3 Influence of Expectations

Expectations are not recognized as part of 55 concepts related to human aspects of software engineering that were found in a literature review on Behavioral Software Engineering [LFW15]. Correspondingly, only limited research exists about expectations in the area of software engineering. This research is centered around either managing expectations in requirements engineering [JS04] or handling them in regards to software users [Pet08].

Jørgensen and Sjøberg [JS04] analyzed the impact of customer expectations on software development effort estimates. They conducted a controlled experiment with 38 students and 12 professionals, that had to estimate the effort needed to develop a specified software system. The authors split the participants into three groups, a control group, a high customer expectation group (HIGH), and a low customer expectation group (LOW). The control group received no further information besides details about the software to be built. In contrast, the HIGH group was given specific customer expectations of 1000 work-hours, while the LOW group received 50 work-hours as an estimate. Both groups were explicitly told to not let these customer expectations impact their estimate. Nevertheless, the results showed, that the HIGH group estimated the necessary work-hours significantly higher (404 for students, 632 for professionals) than the LOW group (77 for students and professionals) and the control group (224 for students, 176 for professionals). Furthermore, the authors evaluated the awareness of this effect and found it to be low. They attribute the impact of the customer expectations to the anchoring effect and warn practitioners to be aware of the problems of using customer effort estimations which potentially contribute to the oftentimes underestimation in industrial software projects.

In regards to user satisfaction of information systems research is concerned with the importance of managing user expectations [Pet08]. Inappropriate expectations can have a downstream effect on satisfaction as well as the usage of users. Hence, researchers apply expectation-confirmation theory to explain the outcomes of improper user expectations. As a result, different strategies to manage expectations already early on in development were proposed to minimize expectation disconfirmation events.

Going outside the field of software engineering, Boot et al. [BSSS13] showed the importance of measuring expectations to properly distinguish between a control group and treatment groups in placebo research. They believe, that "this failure to control for expectations is not a minor omission — it is a fundamental design flaw that potentially undermines any causal inference" [BSSS13, p. 445]. To demonstrate this, the authors analyzed interventions through video games that claim to improve cognition. Studies in this research area compare groups playing action games with a control group playing slower-paced, non-action games and measure their perceptual or cognitive abilities afterwards. In contrast, Boot et al. explicitly measured the expectations of both groups before a possible intervention through two survey studies with 200 participants each. The participants first watched a video of the game they will play (action or control game). Then they learned about the tasks used as outcome measures and had to specify if they believe that their performance would improve in the task as a result of playing the game they watched earlier. The authors found, that participants expected improvement in outcome measures specific to the game they saw. Therefore, participants precisely expected the improvements that are shown in video game intervention research. These results were obtained after only 30 seconds of exposure to a video of the game and independent of prior knowledge of the benefits of game training. Accordingly, Boot et al. believe, that the current research designs do not permit causal conclusions about the effectiveness of game training. Furthermore, the authors found similar patterns in other research such as the benefits of brain-fitness programs, memory exercises, and daily writing. They conclude by highlighting the importance of measuring expectations in research design when comparing different groups.

Since expectations are an integral part of everyday life guiding effective behavior, we believe in accordance with Boot et al. [BSSS13], that it is important to explicitly measure expectations. This allows a better understanding of why experimental groups differ. In our survey, we use a similar design to Jørgensen and Sjøberg [JS04] with three groups to compare the impact of an understandability metric: a control group, a group with suggested low understandability score, and a group with a suggested high understandability score. However, we explicitly measure expectations before the outcome of a subjective code understandability assessment. As a result, we obtain more explanatory power of potential differences which allows us to correlate expectations with motivation and affect. Furthermore, following expectation-confirmation theory we additionally measure expectations after the task to analyze the consequences of differences between expectations before and perceived outcomes after.

3.4 Motivation and Expectancy

Motivation research in the domain of software engineering has been primarily focused on job retention and secondly on motivation in teams as well as in more recent years motivation in open source software (OSS) projects [BBH+08; FGS+11]. Both systematic literature reviews report on the important influences of motivation regarding productivity, software quality, and overall project success [BBH+08; FGS+11]. Expectancy theory was among the least used classical theories in motivation research in software engineering [HBB+09] while self-determination theory is gaining traction as a result of studying OSS projects [FGS+11]. Still, expectancy theory has been successfully employed in various studies to explain the impact of developers' belief for motivation [FGS+11; HBB+09]. In terms of motivating factors, researchers found identification with a task through clear goals, a purpose, and a personal interest [BBH+08] as well as good self-image and learning aspects through self-development [FGS+11] to be important. However, we are not aware of research regarding the motivation of developers towards specific software engineering activities or tasks like bug fixing, refactoring, understanding code, and what role belief in one's abilities plays in such context. Therefore, we present work studying the relationship between expectations and motivation in related activities of problem-solving and learning programming.

Eseryel et al. [ELI+14] investigated problem-solving outcomes of game-based learning as a result of motivation in high school students. For one year, students played an informational massive multiplayer online game (MMOG) and the researchers assessed motivation as well as problem-solving skills before and afterwards. The authors found, that self-efficacy positively predicts the engagement of students indicated by an increase in effort and persistence. Self-efficacy is one's belief in their ability to succeed at some specified goal and therefore very similar to our use of success expectations regarding the understanding of code. In addition, Eseryel et al. showcased the link between motivation, engagement, and problem-solving competencies where motivation determines engagement which in turn determines problem-solving competence. Interestingly, they found a negative relationship between interest and competence with engagement which they tracked to students' unfulfilled expectations after they saw that the game was not as fun as traditional MMOPGs. This represents another example of the expectancy-confirmation theory highlighting the importance of managing expectations.

In an effort to examine motivation towards learning programming skills, Law, Lee, and Yu [LLY10] conducted a study with 365 undergraduate students taking computer programming courses. The results of their questionnaire show, that the intrinsic factor 'individual attitude and expectation' is strongly motivating and strongly correlates with efficacy. Therefore, students who believe to be successful in learning programming are also more motivated to engage in corresponding actions and more confident in their abilities.

Continuously, research shows the importance of motivation in software engineering [BBH+08; FGS+11]. Nevertheless, the understanding of how developers are motivated appears to advance rather slowly [FGS+11]. Therefore, we aim to contribute to this process by investigating how expectations about understanding a code snippet relate to motivation of engaging with that code snippet. In addition, similarly to Eseryel et al. [ELI+14], we analyze how motivation changes before and after a task of understanding code and what impact expectations have in this instance.

3.5 Affective States and Software Developers

It has long been recognized, that affective states influence cognitive processing and therefore performance in humans [BG07]. Developing software is an intellectual, problem-solving as well as creative process [GWA15a]. Therefore, researchers called for a deeper understanding of the human aspect of affect in regards to developers by employing empirical methods and validated measurements [FTAS08; GWA15b]. As a result, this topic has gained attention in academia and industry in recent years [NS19]. The theoretical foundation of affect and explanations of measurement instruments, that are used in the research described below, are defined in Section 2.5.

In an effort to develop a theory of affect regarding developers' performance Graziotin [Gra16], with the help of various colleagues, conducted multiple studies, surveys, interviews, and experiments to analyze the influence of affect on developers' performance in different contexts. Graziotin, Wang, and Abrahamsson [GWA14b] explored the relationship of affective states, creativity, and problem-solving performance of developers through a study with 42 participants. Each participant completed the Scale of Positive and Negative Experience (SPANE) before a creativity task, which involved creating captions for six pictures. Furthermore, the participants had to answer the SPANE questionnaire again, before solving a version of the Tower of London task measuring problem-solving skills. As result, the authors found, that the happiest programmers are more productive in regards to problem-solving performance. In terms of creativity, Graziotin, Wang, and Abrahamsson found no significant results. As part of an analysis of factors that influence the performance in solving coding challenges, Wyrich, Graziotin, and Wagner [WGW19] showed, that sad developers as measured by SPANE perform significantly worse. These results indicate the positive influence of positive affective states on performance, as well as the negative influence of negative affective states. However, Wyrich, Graziotin, and Wagner only found a weak relationship between positive affective state and increased performance, which could be a result of the unique circumstances of coding challenges in contrast to typical software development.

To evaluate correlations between affect dimensions and self-assessed productivity (sPR) Graziotin, Wang, and Abrahamsson [GWA14a] conducted a real-world experiment with eight developers. The authors studied the participants for 90 minutes while they worked on individual projects. They measured the affective state of the developers in terms of valence, arousal, and dominance through the Self-Assessment Manikin (SAM) as well as their sPR every ten minutes. As a result, Graziotin, Wang, and Abrahamsson developed a model that can express 38% of the deviance of sPR based on the differences in affective state. Furthermore, they showed, that valence and dominance are positively correlated with sPR.

Graziotin, Wang, and Abrahamsson [GWA15a] developed their explanatory theory of the impact of affects on programming performance by observing and interviewing two developers during development. The authors studied the two developers over a semester-long real-world project. With the results, they build a model compromising the concepts of events, affects, attractors, and focus which relate to the corresponding programming performance. Events can be work- or non-work-related, public or private, and trigger affective responses. If events become important and a priority for developers they are called attractors, which then encompass the cognitive system of the developer as the driving force for their behavior. Affects and attractors influence the focus of programmers relating to the code and further task or project goals. This focus then impacts the programming performance in the end. The authors found that positive affect leads to increased performance, while negative affect shows the opposite relationship.

In a series of analyses, Graziotin et al. [GFWA17] evaluated a large-scale survey with 1318 participants to explore the causes and effects of happiness and unhappiness of developers on their work. The survey included demographic questions, SPANE for affective states, and two open-ended questions about how affect influences the work of developers in positive and negative ways. The authors discovered that software developers are a slightly happy population in general, happier than all comparable populations they found in the literature. In terms of causes for negative affect, the results suggest that external causes are more prevalent, more precisely they are reported up to four times as much as internal causes. For internal causes, most important are being stuck at problem-solving, feelings of inadequate skills or knowledge, and personal issues. On the other side, external causes are dominated by bad code quality and coding practices, under-performance of colleagues, imposed constraints on development, and time pressure. As for the consequences of negative affective states, the authors observed low cognitive performance, mental unease or disorder, low motivation, and work withdrawal to be the most expressed ones [GFWA18]. Furthermore, participants reported low productivity, delays, and broken flow as external consequences. In contrast, happy developers show high cognitive performance, high motivation, perceived positive atmosphere, higher self-accomplishment, high work engagement and perseverance, higher creativity, and higher self-confidence. For external consequences of happiness, the authors found high productivity is the most frequently listed one, followed by expedition, sustained flow, increased collaboration, and increased process adherence as well as high code quality as a consequence for the software artifact under development.

This research shows the importance of studying affect in relation to software developers. It highlights the influence of affect on performance, code quality, and self-image as well as related concepts such as motivation. Following these results, we evaluate whether understandability metrics and expectations elicited as a result of them impact affective states. We use SAM similarly to Graziotin, Wang, and Abrahamsson [GWA14a] to measure affect before and after a code understandability task. Furthermore, we explore the consequences of unfulfilled expectations on affective states.

Chapter 4

Methodology

This chapter outlines the method and steps we took to develop the research design. We report the research questions and the experiment design, participants, materials, and experiment procedure to answer them. Furthermore, we describe our efforts to mitigate threats to validity. At the end, we define our variables and hypotheses as well as the procedure we use for the analysis of the results. We follow the guidelines of reporting experiments in software engineering [JCP08] but utilize their advice to change the order and combine sections if it fits better.

4.1 Research Questions

This work aims to analyze how an understandability metric influences expectations about understanding a code snippet and consequently impacts motivation and affective states in regards to the task of understanding the code snippet. Therefore, we answer the following research questions:

RQ1 How does a metric value of code understandability affect expectations about understanding the code?

RQ2 How do source code understandability expectations influence motivation to engage with the code?

RQ3 How do source code understandability expectations influence affective states?

RQ4 How do the differences between expectations and perception of source code understandability influence motivation and affective states?

RQ5 Does the value of a presented code understandability metric influence the subjective assessment of code understandability?

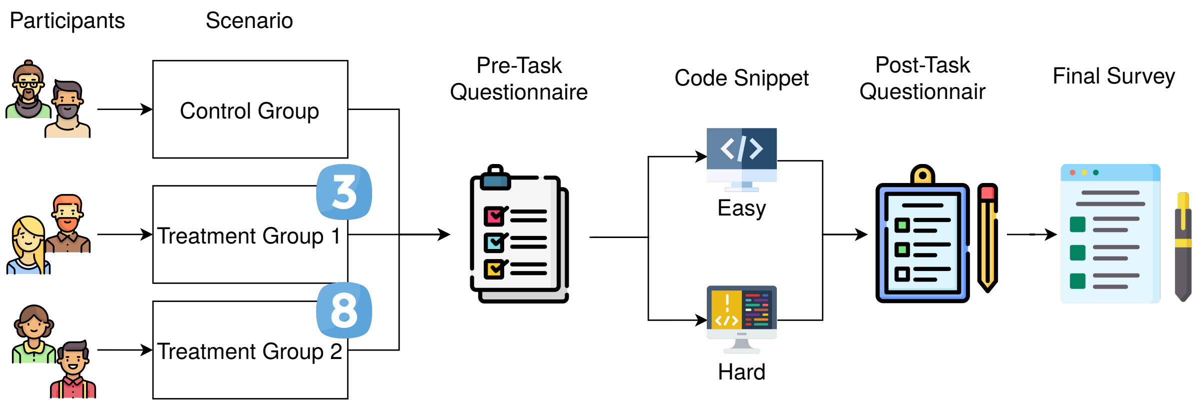


Figure 4.1: Experiment design using three randomly assigned groups with different scenarios of suggested understandability and two code snippets [Fla].

4.2 Experiment Design

In order to answer the research questions, we conduct an experiment through an online survey. Before the experiment starts, we inform participants about the upcoming task of judging the understandability of a presented code snippet with additional questions about expectations, motivation, and affective states before and after this task.

The experiment design can be seen in Figure 4.1. It is a between-subjects design and consists of three groups to which participants are randomly assigned to. The scenario description in the beginning starts with a rough outline of the upcoming code snippet in terms of expected lines of code and functionality implemented. Additionally, one group is told, that the upcoming code snippet will be easy to understand, which is denoted as a 3 out of 10 on a scale from 1 to 10 where 1 is very easy to understand and 10 is very hard to understand. The other group receives the same scenario description but is told that the upcoming code snippet will be hard to understand, denoted as an 8 out of 10 on the same scale. Both groups are presented with the same story of an expert system that rated the upcoming code snippet based on multiple metrics as an explanation for this judgment. Lastly, the third group acts as the control group receiving no assessment of the upcoming code snippet. Based on the scenario description all participants answer the same questionnaire about expectations, motivation, and affective state. Afterwards, a task description about the upcoming code snippet and expected assessment by the participants follows. For the two treatment groups, we repeat the judgment from the expert system at this point in terms of 3 out of 10 and 8 out of 10 respectively. Then participants are randomly assigned to one of two code snippets where the task is to judge the understandability on the same scale the expert system uses from 1 (easy to understand) to 10 (hard to understand). After participants assessed the code snippet, they answer the almost same questionnaire involving expectations, motivation, and affective state. At the end, a short survey with demographic questions and a segment about the personal opinion of the participants regarding the influence of understandability metrics finishes the survey.

With this design, we explore differences in measured expectations between the groups based on the initial scenario description involving two different understandability metrics or no metric. The scenario description specifies some rough information about the code snippet with the purpose that participants are able to answer the expectation, motivation, and affective state questions. This is especially required for the control group because otherwise, participants in this group would have nothing to base their answers on. For the other two groups, we use 3 and 8 respectively for the expert systems judgment of the upcoming code snippet as opposites on the 10 point scale to have a significant difference between the groups while still being potentially accurate when seeing the code snippets. Based on this information from the first questionnaires, the expectations can then be correlated with the measured motivation and affective state. Furthermore, we use two code snippets to create differences between the expectations as a result of the presented scenario and the perception of the actual code to answer RQ4. Therefore, one code snippet is rather easy to understand, while the other is considerably harder to understand. As a result, we can analyze how participants to which we suggest that they will see a hard code snippet react to looking at an easy code snippet and the other way around. In addition, we can evaluate the differences in subjective understandability judgments between the three groups for two code snippets that vary in difficulty.

4.3 Participants

To gather participants we invite potential suspects through E-Mail, modern messenger services, and social media in a convenience sampling strategy. Part of the participants are software students which are mostly in masters degree courses. Furthermore, we use contacts with multiple software companies to additionally invite IT professionals. We message each contact by their preferred service and encourage them to further distribute the survey with colleagues and others who are eligible to participate. Additionally, we post the survey on our private Facebook page and on /r/SampleSize[1], a subreddit dedicated to posting and participating in surveys for academic or casual purposes.

The only requirement for participation is a basic understanding of the Java programming language since the code snippets we use are in Java. We encourage participation with the low amount of time commitment of 10 to 15 minutes for the whole survey.

[1]https://www.reddit.com/r/SampleSize/

Furthermore, the author pledges to donate 5€ to a good cause for every participant that completes the survey. The subjects can choose between three charity projects involving different topics or they have the option to evenly split the donation between the three projects. Additionally, following the goal-setting theory of motivation [LL02] we invite participants with a clear goal of judging the understandability of a given code snippet. Setting such a specific and challenging goal can lead to higher effort and persistence, which is desirable for survey completion. Therefore, we hope to gain the interest of developers that want to show off their ability to understand code.

Participants' consent to the data and privacy policy is obtained on the landing page of the survey including an explanation of the anonymous nature of the survey to ensure participants that no recognizable data is required.

4.4 Materials

For our experiment participants have to complete an online survey. We use LimeSurvey[2] to create and provide our survey. LimeSurvey grants a free to use community edition of its complete software, which can be installed and run on any server. Furthermore, it supports all necessary functionality including various question types, the grouping of questions, randomization, conditioned paths through the survey, anonymous data collection, timing statistics, and easy results exports. Additionally, LimeSurvey provides the option to create individual question types and insert any customization of themes, icons, and functionality. Therefore, we set up our survey in LimeSurvey, host it on a university internal server, and adapt it to our needs.

The final survey consists of eight pages. They are split up into three sections of pre-task, task, and post-task. The pre-task section covers two pages displaying the scenario description and the pre-task scales of expectation, motivation, and affective state. Afterwards, the task description is on a separate page before the code snippet and the corresponding scale to judge its understandability are presented. Identical to the pre-task section, the post-task section is split over two pages displaying the adapted versions of the same scales as before. Afterwards, participants have to answer demographic questions in terms of age, gender, current occupation, programming experience, and the frequency of understanding code being part of their job as well as the frequency of use of static analysis tools. We measure programming experience threefold, following Siegmund et al. [SKL+13] who evaluated programming experience questions and built a model through a controlled experiment. Their analysis showed, that self-estimated

[2]https://www.limesurvey.org

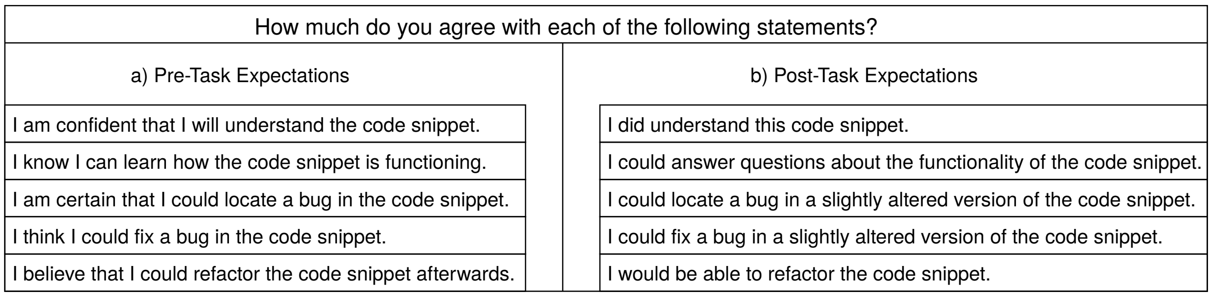
programming experience is a good indicator. Therefore, we use their proposed 1 to 10 scale regarding general programming experience, a comparison with 10 year experts on a 5-point Likert scale, and experience in OO-programming (Java) on a 5-point Likert scale to measure experience. In the original paper, the comparison to experts involved experts with 20 years of programming experience. However, the authors do not provide a reason for the choice of 20 years. We use 10 years instead because we believe it creates a more evenly and meaningful distribution since the amount of developers who think they are (way) better than 20 years experts is rather low. On the last page, we reveal the intention of our survey and ask the subjects to provide their personal opinion on the topic of understandability metrics potentially influencing their motivation and emotions[3]. To this end, we ask our participants to envision a scenario of working with a static analysis tool providing an understandability metric. Then, they have to indicate how a bad (good) score would impact their motivation and emotions. Lastly, we ask them whether they believe that code understandability metrics are accurate.

4.4.1 Expectation Scales

In order to answer research questions RQ1-4, we need to measure participants' expectations about understanding the code snippet. First, we require the expectations before seeing any code snippet based only on the scenario description. Additionally, for RQ4 we have to gather the perception of the code after seeing it to be able to measure the difference between pre-task and post-task.

This is typically achieved through dedicated expectation scales. Since there does not exist a validated expectation scale in regard to understanding code as far as we know, we have to develop our own scale. We study the scales mentioned in Section 2.3.2 to learn how such scales are constructed, how to phrase them, and how to measure them in the end. Expectation scales consist of multiple items that are rated on a Likert scale and are then combined in the end towards on expectation value. To create multiple items reflecting the expectation of understanding code we adopt the methods researchers use to measure code understandability. These methods are presented in Figure 2.1 and consist of comprehension questions, subjective rating, locate bug, modifying task, cloze test, fix bug, and recall. Since researchers use these methods as measurement instruments, we believe they best represent how developers could think about whether they understand a code snippet. From this list, we use the ones which someone can reasonably answer without actually seeing any code based only on a rough description.

[3]In our survey, we always use the word emotion instead of affective state, because it is typically used in everyday contexts, shares a widespread understanding, and does not need a theoretical introduction beforehand.



How much do you agree with each of the following statements?

a) Pre-Task Expectations	b) Post-Task Expectations
I am confident that I will understand the code snippet.	I did understand this code snippet.
I know I can learn how the code snippet is functioning.	I could answer questions about the functionality of the code snippet.
I am certain that I could locate a bug in the code snippet.	I could locate a bug in a slightly altered version of the code snippet.
I think I could fix a bug in the code snippet.	I could fix a bug in a slightly altered version of the code snippet.
I believe that I could refactor the code snippet afterwards.	I would be able to refactor the code snippet.

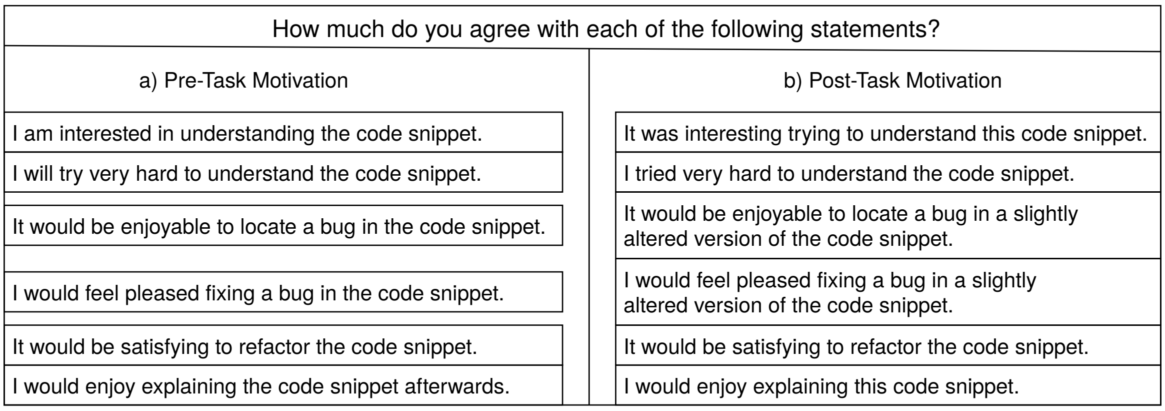
Figure 4.2: Expectation scales used to measure expectations before the task of understanding a code snippet (left side) and after the task (right side).

The results can be seen in Figure 4.2. The expectation scale consists of five statements, where each statement is rated on a 5-point Likert scale ranging from strongly disagree to strongly agree. On the left side, noted with a), the statements refer to the upcoming code snippet, while on the right side, noted with b), the statements are phrased in the past tense regarding the code snippet participants saw before. Furthermore, we use the phrase "in a slightly altered version" to signal participants to imagine the same code snippet with a possible bug inserted.

4.4.2 Motivation Scales

In a similar manner to expectations, we want to capture the motivation participants have in regards to understanding the code snippet. As we reviewed in Section 2.4.1 different methods exist to measure motivation. However, since we use an online survey only some apply to our case. Therefore, we choose the most prominent and easily administrable one in a motivation scale. We discussed but ultimately rejected the idea of employing a second instrument for motivation using the time participants spend on understanding the code snippet. The options included utilizing a free choice variation where we would set a predetermined amount of time available for understanding the code and participants can choose to increase the time after it expired. However, we could not be sure if we could attribute differences in time used between participants to motivation or other factors. Furthermore, time limits could induce stress and anxiety which would impact our measurements for affective states.

We use a motivation scale in line with our expectation scale discussed above. Screening the literature, we are not able to find a validated scale applicable to our context. Most often these scales can not be administered only concerning one specific task and especially not two times in a short amount of time, like before and after the task. Even though many scales in different contexts exist, they usually measure motivation in a broader sense like motivation to study science or math. Therefore we develop our own



How much do you agree with each of the following statements?

a) Pre-Task Motivation	b) Post-Task Motivation
I am interested in understanding the code snippet.	It was interesting trying to understand this code snippet.
I will try very hard to understand the code snippet.	I tried very hard to understand the code snippet.
It would be enjoyable to locate a bug in the code snippet.	It would be enjoyable to locate a bug in a slightly altered version of the code snippet.
I would feel pleased fixing a bug in the code snippet.	I would feel pleased fixing a bug in a slightly altered version of the code snippet.
It would be satisfying to refactor the code snippet.	It would be satisfying to refactor the code snippet.
I would enjoy explaining the code snippet afterwards.	I would enjoy explaining this code snippet.

Figure 4.3: Motivation scales used to measure motivation before the task of understanding a code snippet (left side) and after the task (right side).

motivation scale using the same concepts of understandability measures we use in the expectation scale. We focus on intrinsic motivation since any extrinsic motivation factors are impacted by or do not work as a result of an anonymous survey. Furthermore, we are more interested in intrinsic motivation since it determines whether a developer enjoys engaging with the code instead of doing it as a requirement of their job. The main inspiration for our scale is the Intrinsic Motivation Inventory (IMI) [DR20] and other scales based on it. Through inspecting them we gather phrases and typical verbs used to describe motivation towards an activity. We mainly adopt statements from the interest and enjoyment category of the IMI since it is the most direct measure of intrinsic motivation. In addition, the other categories are impacted by our survey setting. The resulting motivation scale is presented in Figure 4.3. We use the same statements before the task, displayed on the left in a), and alter them accordingly for after the task, presented on the right in b). Again, each statement is answered on a 5-point Likert scale ranging from strongly disagree to strongly agree.

4.4.3 Affect Measurement

For the measurement of affective states, we use the pictorial method of the Self-Assessment Manikin (SAM) [BL94]. It is widely employed and validated in the literature and has been successfully used previously in related work in the domain of software engineering to measure affective states. Additionally, this instrument can be administrated multiple times which is necessary for our pre-task to post-task comparison. SAM is depicted in Figure 2.3. Through SAM we gather affective states in the dimensions of valence, arousal, and dominance. Valence characterizes how happy or unhappy an individual is. Arousal represents the activation of a person in terms of being calm or excited. Lastly, dominance describes the level of autonomy or self-control one feels.

Combining them provides us a picture of the affective state an individual currently is in.

We adapt a 5-point selection question in LimeSurvey to display the five SAM figures for each dimension where every participant is asked to select the figure that best represents them. Furthermore, we supply the instructions for each dimension as help text from the technical manual [LBC+97] underneath each set of figures. We slightly adapt the instruction texts to reflect the ordering of the figures we use from left to right and the setting of the figures in terms of them being in a row. Furthermore, we only use the parts of the instructions which are necessary to understand the figures. This helps participants to understand the figures while not being too long. In a preliminary run of the survey, we find that this help text was especially necessary for the dominance dimension since it is not as intuitive as the other two.

4.4.4 Code Snippets

For the experiment, we use two different code snippets. We want code snippets that can be understood without domain knowledge and without any complicated or unknown language constructs. Furthermore, they should be not too long in order to fit on the survey but still challenging to make it worthwhile to try to understand and judge them. As a result, we make the survey accessible to a wide range of developers because they only need a basic level understanding. Additionally, it fits into our goal of a compact, not too long, survey that is still fun to participate in.

For our experiment design, one code snippet has to be easy to understand while the other is hard to understand. In the following, we abbreviate these two into easy code snippet and hard code snippet. Since every programmer has a slightly different idea of how understandable code looks like and since there does not exist an objective measurement of understandability there is no clear way on how to select these code snippets. Therefore, we pre-select code snippets based on Cognitive Complexity and conduct a preliminary survey run with eight developers to gather their assessment of the code snippets. Matching our code snippet criteria we follow previous researchers [FMAA18; Pre20] and select code snippets from the Apache commons-lang StringUtils[4] project. These methods manipulate Strings and therefore use basic Java constructs and do not require domain knowledge.

In a first step, we select two code snippets fitting the easy to understand and hard to understand type based on a difference in Cognitive Complexity and the author's

[4]https://github.com/apache/commons-lang/blob/master/src/main/java/org/apache/commons/lang3/StringUtils.java

subjective assessment. The preliminary study results in accepting the easy code snippet and rejecting the hard code snippet since it was not perceived as difficult enough. Therefore, we conduct a second round with five programmers that had to select the most difficult to understand code snippet amongst four choices. In addition to three code snippets with a high Cognitive Complexity from the StringUtils project, we provide one difficult string challenge solution found on the internet. Four out of five programmers agree on the same code snippet to be the most difficult which is the one we use in our survey. The final selected code snippets are displayed in Listing 4.1 which is the easy code snippet and Listing 4.2 which is the hard code snippet.

```
/**
 * Checks if the String contains mixed casing of both uppercase and lowercase characters.
 *
 * @param str the String to check, may be null
 * @return true if the String contains both uppercase and lowercase characters
 */
public static boolean isMixedCase(final String str) {
    if (isEmpty(str) || str.length() == 1) {
        return false;
    }
    boolean containsUppercase = false;
    boolean containsLowercase = false;
    final int sz = str.length();
    for (int i = 0; i < sz; i++) {
        if (containsUppercase && containsLowercase) {
            return true;
        } else if (Character.isUpperCase(str.charAt(i))) {
            containsUppercase = true;
        } else if (Character.isLowerCase(str.charAt(i))) {
            containsLowercase = true;
        }
    }
    return containsUppercase && containsLowercase;
}
```

Listing 4.1: Code snippet we use in our experiment as the "easy to understand" variation [Apa].

For the easy code snippet, we change the CharSequence in the original code to a String because it does not alter the functionality but makes it more accessible to inexperienced developers. Furthermore, we adapt the method comment to only reflect the necessary information.

The hard code snippet originally prints the result in the end which we change to a return statement matching the easy code snippet. Additionally, we add the method comment which does not exist in the original version.

```java
/**
 * Find the longest palindromic substring within a string.
 * A palindrom is a word that reads the same backwards as forwards, e.g. madam.
 *
 * @param str1 the String to search in
 * @return the longest palindromic substring
 */
public static String longPalSubstr(String str1) {
    int n = str1.length();
    boolean table[][] = new boolean[n][n];
    int mLength = 1;
    for (int i = 0; i < n; ++i)
        table[i][i] = true;
    int strt = 0;
    for (int i = 0; i < n - 1; ++i) {
        if (str1.charAt(i) == str1.charAt(i + 1)) {
            table[i][i + 1] = true;
            strt = i;
            mLength = 2;
        }
    }
    for (int k = 3; k <= n; ++k) {
        for (int i = 0; i < n - k + 1; ++i) {
            int j = i + k - 1;
            if (table[i + 1][j - 1] && str1.charAt(i) == str1.charAt(j)) {
                table[i][j] = true;

                if (k > mLength) {
                    strt = i;
                    mLength = k;
                }
            }
        }
    }
    return str1.substring(strt, strt + mLength);
}
```

Listing 4.2: Code snippet we use in our experiment as the "hard to understand" variation [w3r].

4.5 Survey Procedure

On the landing page of the survey, participants have to consent to the data and privacy policy. They are informed about the different sections of the survey, the handling of the data, and the intent to publish the results. Furthermore, they are given the option to stop and end the survey at any time without drawbacks and automatic deletion of previous answers. In order to ensure the privacy of the data, the survey is hosted on a university intern server. Furthermore, the survey is fully anonymous. We do not collect any IP address, timestamp, or other information that could recognize an individual participant.

After consenting to the data and privacy policy, participants are presented with the following scenario description depending on the group they are randomly assigned to:

> "You will look at a Java code snippet in a bit. The only information that we provide you about the snippet is the following: The code snippet is between 20 and 30 lines long and tests a String, a sequence of characters, for a criterion.
>
> (We developed an expert system to rate the understandability of code based on multiple metrics. This system rated the code snippet you will look at in a few moments as an (3/8) out of 10. The system uses a scale from 1 to 10, where 1 is very easy and 10 is very hard to understand.)
>
> In the following questions, you will be asked about your expectations. Answer based on only the information we provided you."

Participants in the control group are shown everything except the part in brackets while the other two groups additionally see the part in the brackets and a 3 and 8 respectively for the metric. Afterwards, participants answer the expectation scale, motivation scale, and SAM which are presented in the sections before. The scenario description is repeated on the second page which presents the SAM figures. Here, participants are asked to select the figure that best represents them on each dimension regarding their feeling about the upcoming task of understanding the code snippet. On the next page the instructions about the upcoming code snippet are shown as follows:

> "Next, you will look at the Java code snippet. Your only task is to judge its understandability on a scale from 1 to 10, where 1 is very easy to understand and 10 is very hard to understand. The code is fully functioning and bug-free.
>
> (You will rate the code snippet on the same scale that is used by our expert system, which rated it as an (3/8) out of 10. We are interested in what you think.)
>
> You will have unlimited time to look at the code snippet. Feel free to rate the code snippet whenever you think you have an adequate impression to judge its understandability."

Again, the control group sees the text without the part in brackets while the other two groups additionally see the bracketed part in accordance with their scenario. Following this, the participants see one of the two code snippets to inspect its understandability and are asked to rate the understandability on the provided scale. Afterwards, the subjects complete SAM again as well as the post-task versions of the expectation scale and motivation scale. This time the description asks them "how they felt about trying to understand the code snippet" for SAM and how they see the expectation and motivation statements now after seeing the code snippet. The next page consists of the demographic questions which are described in Section 4.4. On the last page, we reveal our intentions with this survey in terms of "investigating the influence of metrics on the motivation and emotions of software developers". We explain that participants were presented with different scenarios in the beginning and ask them about their personal opinion concerning the topic of the survey. This last page additionally includes a section where participants can select which donation project they want to support and an open text segment for further comments. At the end, participants are thanked for their participation as well as encouraged to further distribute the survey.

4.6 Mitigating Threats to Validity

This section describes our efforts in mitigating potential threats to validity. We consider concerns based on the book of Wohlin et al. [WRH+12] about experimentation in software engineering and the catalog of 39 confounding parameters in regards to program comprehension experiments [SS14]. Since there exists considerable overlap between the two we report the terminology used by Wohlin et al. in that case.

Independent of specific threats, we conduct a preliminary survey with eight developers to test our design and find potential problems in the descriptions, measurement instruments, and experiment flow. As a result, we adapt the scenario description to create a more even starting position for all three groups to secure that differences are a result of the treatments. Furthermore, we change a code snippet to better reflect our intentions of a hard code snippet. In the end, we conclude, that the design works for the goal of our experiment and the chosen instruments can be easily understood by participants and measure the desired constructs.

4.6.1 Mitigating Threats to Conclusion Validity

These threats are concerned with issues of drawing the correct conclusions from the treatments and observed outcomes. For this category, most issues relate to the use of the right statistical methods which is discussed in Section 4.8. Additionally, we use validated measurement instruments wherever possible and otherwise construct instruments based on validated ones to ensure the reliability of measures [WRH+12]. Furthermore, the use of an online survey guarantees that each participant receives exactly the same treatment (*reliability of treatment implementation* [WRH+12]).

4.6.2 Mitigating Threats to Internal Validity

Internal validity relates to the causality of the independent variables regarding the outcome. To ensure the validity of our treatments we use a control group which mitigates most concerns with group designs [WRH+12]. Additionally, all groups go through the same survey with identical instruments. The only differences occur in the scenario and task description as presented in Section 4.5. Therefore, we establish that potential discrepancies are a result of the different treatments.

Participants have to answer the expectation scale and motivation scale two times: before and after the task. Therefore, to mitigate unintended learning and encourage participants to engage with and read the statements again the second time around, the ordering of the statements for both scales at both times are randomized (*testing* [WRH+12]).

We design our survey to be relatively short in terms of time in order to mitigate problems of tiredness or boredom (*maturation* [WRH+12]). Furthermore, there are no time restrictions on the survey nor the task of rating the understandability of the code snippet to circumvent the induction of stress or time pressure. Additionally, the use of an online survey mitigates social threats in terms of participants learning about the treatment or compensation of another group through social interaction which influences their performance [WRH+12]. However, our survey is distributed by other people than the authors through the encouragement of sharing it which could introduce problems where people share too much information after they participated themselves. To mitigate this threat, we provide an example invitation and highlight the information which can be shared beforehand. Furthermore, as a result of the anonymity of the survey, we can not control the truthfulness with which participation occurs and subjects could participate multiple times. In addition, we use a small donation incentive for participation. We do not believe, that this encourages participation in mischievous ways since there is no personal benefit to them. However, to further mitigate these potential problems, we conduct an internal review of the survey responses based on timing statistics to exclude participants that use unreasonably low amounts of time to answer the survey.

For the survey, we use an established software tool in LimeSurvey to automatically collect, store, and provide the results of all measurements as well as timing statistics which ensures no manual errors in data collection (*instrumentation* [WRH+12]).

4.6.3 Mitigating Threats to Construct Validity

Construct validity is concerned with the theories behind the experiment and its design. We use proper theory of other scientific fields for the involved concepts in expectancy, motivation, and affect in order to ensure the validity of the constructs in use. In addition, two treatments and two code snippets are used to more extensively represent the constructs under analysis (*mono-operation bias* [WRH+12]).

Furthermore, participants are randomly assigned to the different groups. The subjects do not know of the existence of different groups. Additionally, to cover the intentions of the experiment we use an appropriate story of participants having to judge the understandability of a code snippet (*hypothesis guessing* [WRH+12]). This story is reinforced by the scenario description. Moreover, for the two treatment groups, the introduction of the expert system might lead them to think that we want to validate or compare their judgment to this system. Therefore, the stated goal relates to the experiment but does not reveal the full picture.

With the use of an online survey for our experiment we automatically circumvent most social threats to construct validity [WRH+12]. These include influences an experimenter can have on participants through their expectations regarding the experiment which are called experimenter bias and demand characteristics [KDL+12] where participants guess how to act in order to be a "good" subject based on how the experimenter reacts. Furthermore, the anonymous nature of the survey mitigates potential evaluation problems, where participants act and answer differently as a result of feeling observed (*evaluation apprehension* [WRH+12]).

4.6.4 Mitigating Threats to External Validity

Threats to external validity relate to the generalizability of the results. We invite both students and IT professionals to gather a balanced account of software developers (*interaction of selection and treatment* [WRH+12]).

Furthermore, we use a realistic setting for our experiment (*interaction of setting and treatment* [WRH+12]). Our scenario description includes general information about the length and functionality of the code snippet. The understandability metric in the treatments describes the relative understandability of the code snippet. This is similar to a warning of a static analysis tool that typically involves a level of seriousness of the potential problem. In addition, developers usually have at least a rough idea of the corresponding part of code. Furthermore, we use real code snippets instead of creating them ourselves. This includes a method comment describing the functionality of the code snippet. Moreover, it ensures that typical variable names are used as well as a standard implementation of the function. Additionally, we mitigate confounding factors of individual knowledge which include ability, domain knowledge, education, and familiarity with tools by working with code snippets that do not require any domain knowledge as well as ones that only use basic language types and constructs [SS14]. The code snippets are presented in a separate box with common indentation and highlighting which eliminates potential bias through a specific development environment. No restrictions exist to use the web if participants do not know or understand something which reflects real life.

4.7 Hypotheses, Parameters, and Variables

For the experiment, we use three different groups: a control group (C), a group to which we suggest an understandability metric value of 3 (M_3), and a group to which we suggest the metric value of 8 (M_8). Additionally, we work with two code snippets of different understandability difficulty which we call easy and hard respectively. These are the independent variables of our experiment.

In order to answer our research questions, we measure expectations through five statements which are measured on a 5-point scale. The individual answers to these statements are first added up and then divided by the number of statements. The result is an expectation_sum which is measured pre-task and post-task and ranges from 1 to 5. Similarly, the motivation scale we use consists of six statements that are first combined into one value and then divided by six which results in a motivation_sum for pre-task and post-task with the same range of 1 to 5. For affective states, we use SAM which is measured on a 5-point scale for the dimensions valence (V), arousal (A), and dominance (D). Again, SAM is administrated pre-task and post-task. Furthermore, participants subjectively judge the understandability (sU) of the code snippet on a scale from 1 to 10. Additionally, in order to answer research question RQ4, we measure the difference between expectations pre-task and post-task by calculating: expectation_difference = post-task_expectation_sum − pre-task_expectation_sum. We apply the same concept to the motivation_sum and the SAM dimensions V, A, and D. Therefore, the results are positive when the values are higher after seeing the code snippet and negative when they are lower in the post-task section. These parameters comprise our dependent variables.

In regards to our five research questions we establish the following hypotheses:

Hypotheses for RQ1 For research question RQ1 we use the pre-task_expectation_sum and propose the following hypotheses:

H_{001} There exists no significant difference between the pre-task_expectation_sum of the control group, M_3, and M_8.

H_{011} There exists a significant difference between the pre-task_expectation_sum of the three groups in terms of $M_3 > C > M_8$.

When a static analysis tool displays a metric indicating how understandable a certain part of code is, it produces expectations about that code in a developer. As discussed in Section 2.3 these expectations are built through past experiences with understandability metrics and how easily understandable code looks like in contrast to hard to understand code. Therefore, we expect that participants that are suggested to see an easy to understand code snippet to have higher success expectations in terms of understanding the code snippet than participants in the control group and the group we suggest that they will see a hard to understand code snippet.

Hypotheses for RQ2 To answer research question RQ2 we correlate the pre-task_expectation_sum with the pre-task_motivation_sum and present the following hypotheses:

H_{002} The pre-task_expectation_sum does not significantly correlate with the pre-task_motivation_sum.

H_{012} The pre-task_expectation_sum significantly correlates positively with the pre-task_motivation_sum.

If individuals have high success expectations in regards to a task or goal it generally motivates them to achieve it. Multiple motivation theories are built on this connection (see Section 2.4). These theories show, that the belief of an individual in their own abilities is an important part of motivation. As a result, we expect participants with high expectations to also express high motivation in regards to engaging with the code snippet.

Hypotheses for RQ3 For research question RQ3 we correlate the pre-task_expectation_sum with the pre-task results of SAM in terms of valence, arousal, and dominance and introduce the following hypotheses for the three dimensions:

H_{003} There exists no significant correlation between the pre-task_expectation_sum and the pre-task affective state measurement of valence.

H_{013} There exists a significant positive correlation between the pre-task_expectation_sum and the pre-task affective state measurement of valence.

The expectancy and placebo research discussed in Section 2.3 suggest, that expectations influence the affective state of individuals. Optimistic expectations can produce positive affect. Therefore, we expect a positive correlation between expectations and valence. Furthermore, Graziotin, Wang, and Abrahamsson [GWA14a] showed a positive correlation between valence and dominance and self-assessed productivity. This could indicate a similar correlation for expected performance.

H_{103} There exists no significant correlation between the pre-task_expectation_sum and the pre-task affective state measurement of arousal.

H_{113} There exists a significant correlation between the pre-task_expectation_sum and the pre-task affective state measurement of arousal.

On the other side, arousal did play no role in the correlation of the results from Graziotin, Wang, and Abrahamsson. Therefore, we abstain from any directional hypothesis for arousal.

H_{203} There exists no significant correlation between the pre-task_expectation_sum and the pre-task affective state measurement of dominance.

H_{213} There exists a significant correlation between the pre-task_expectation_sum and the pre-task affective state measurement of dominance.

Graziotin, Wang, and Abrahamsson showcased the influence of dominance in their experiment. However, we lack additional research that supports these results. We suspect a potential positive correlation but we believe the possibility for a negative relationship exists which is why this alternative hypothesis remains directionless.

Hypotheses for RQ4 In regards to RQ4 we analyze correlations between the expectation_difference and the motivation_difference as well as V/A/D_difference and declare the following hypotheses:

H_{004} There exists no significant correlation between expectation_difference and motivation_difference.

H_{014} There exists a significant correlation between expectation_difference and motivation_difference.

H_{104} The expectation_difference does not significantly correlate with the valence_difference.

H_{114} The expectation_difference does significantly correlate with the valence_difference.

H_{204} No significant correlation exists between expectation_difference and arousal_difference.

H_{214} A significant correlation exists between expectation_difference and arousal_difference.

H_{304} Expectation_difference and dominance_difference do not significantly correlate.

H_{314} Expectation_difference and dominance_difference do significantly correlate.

Following expectancy theory and expectancy-confirmation theory, expectations that are not fulfilled can have an increased effect on affective states and future motivation (see Section 2.3). Any perceived event is evaluated in contrast to the expectations an individual held beforehand. As a result, unexpected events lead to more extreme consequences. Correspondingly, we expect to find that differences in expectations and perception from pre-task to post-task lead to differences in motivation and affective states. Furthermore, we expect to observe differences between the combinations of treatment groups and code snippets. Therefore, in addition to these hypotheses, we inspect specifically how participants react that receive the opposite types of scenario and code snippet, for example, M_3 and the hard code snippet in comparison to participants that receive what they expected.

Hypotheses for RQ5 To answer research question RQ5 we inspect how the three groups differ in their subjective understandability assessment of the two code snippets and submit the following hypotheses:

H_{005} There exists no significant difference between the three groups (C, M_3, M_8) and their sU of the easy code snippet.

H_{015} The three groups (C, M_3, M_8) significantly differ in their sU of the easy code snippet.

H_{105} There exists no significant difference between the three groups (C, M_3, M_8) and their sU of the hard code snippet.

H_{115} The three groups (C, M_3, M_8) significantly differ in their sU of the hard code snippet.

In a controlled experiment Preikschat [Pre20] showed, that manipulating an understandability metric significantly and strongly impacts the subjective understandability assessment of participants. According to anchoring research (see Section 2.2) this effect is very robust. Even experts in the field and implausible anchors only mitigate the anchoring effect. Therefore, we expect to be able to replicate the results of Preikschat and find significant differences for our two code snippets between the three groups.

4.8 Analysis Procedure

We export the results from LimeSurvey in an excel format to analyze the data in Python. The 5-point Likert scale data is automatically converted into the corresponding numbers of 1 to 5 and the necessary variables to test our hypotheses are computed including sums and differences. For the statistical analysis and hypothesis testing, we use the standard Python libraries including scpy.stats[5], scikit-learn[6], and statsmodels[7].

There exists lengthy discussion about how to treat Likert scale data in regards to statistical analysis [Ber07; Jam04]. The argument covers the problem of Likert scale data representing ordinal data or interval data. While the controversy continues for single Likert items, which is one statement of a scale, there is no problem treating Likert scales which combine multiple items into one scale as interval data [CP08; Nor10]. Therefore, we consider our expectation and motivation scale to be interval data for statistical analysis. As for SAM, which measures each affect dimension on a 5-point scale, we follow previous researchers and treat it as interval data reporting means and standard deviations [BL94; GWA14a; GWA15b].

In order to evaluate the research questions we follow the typical statistical analysis procedure in verifying assumptions for statistical tests and as a result, using the appropriate parametric or non-parametric test [MB16]. For research question RQ1 and RQ5, we analyze differences between the three treatment groups. Therefore, we use one-way ANOVA if the assumptions of normally distributed residuals and homogeneity of variance are met. Residuals are the difference between the observed value of the dependant variable and the predicted value of the model. A normal distribution is determined through the Shapiro-Wilk test as well as inspection of the Q-Q plot and histogram of the residuals while homogeneity of variance is examined through Levene's test. If the assumptions are not verified we use the non-parametric alternative of the Kruskal-Wallis test. We are aware of the robustness of the one-way ANOVA test especially in regards to normality which we consider if the assumption tests only fail slightly [Nor10]. For the case that the statistical test shows significant results, we apply post-hoc testing to find the groups that differ significantly using the Tukey Honestly Significant Difference (HSD) for ANOVA and pair-wise Mann-Whitney U tests for Kruskal-Wallis.

For research question RQ2, RQ3, and RQ4 we try to find significant correlations between two variables. Therefore, we use Pearson's correlation coefficient when the assumptions of normally distributed variables, linearly related variables, and homogeneity of

[5]https://docs.scipy.org/doc/scipy/reference/stats.html
[6]https://scikit-learn.org/stable/
[7]https://www.statsmodels.org/stable/index.html

variance are met [SBS18]. Furthermore, for RQ2 and RQ3 we employ simple linear regression if a significant correlation exists and the assumptions for linear regression are fulfilled: linearity between variables, mean of residuals is near zero, normality as well as homogeneity of variance of residuals, and no autocorrelation of the residuals [CF14; EA17]. Linearity is assessed through the inspection of scatter plots. Normality is verified through the Shapiro-Wilk test, an inspection of Q-Q plots, and histograms. Homogeneity of variance is measured using Levene's test for correlation and through the Breusch-Pagan test and the Goldfeld-Quandt test as well as the scatter plot of residuals for linear regression. Lastly, autocorrelation is assessed with the Durbin-Watson test. In the case of non-normality for one variable, we employ logarithmic and square root data transformations to check for possible improvements, which are suggested as possible fixes for non-normality [Vet17]. Otherwise, we use Kendall's τ correlation coefficient as the non-parametric alternative. It is more robust and efficient than Spearman's rank correlation [CD10] but the found correlations are significantly smaller than from Spearman's [FN07].

Additionally, we provide the appropriate effect sizes for each test to describe the meaningfulness of results. The strength of these effect sizes are described based on the extended version of Cohen's guidelines in regards to Cohen's d: d(.01) = very small, d(.2) = small, d(.5) = medium, d(.8) = large, d(1.2) = very large, and d(2.0) = huge [Coh88; Saw09]. Since they are widely known, we convert other effect size measures such as Kendall's τ and η^2 into Cohen's d according to Lenhard [Len16] and Walker [Wal03].

A few of our hypotheses like H_{011} and H_{012} would warrant one-tailed hypothesis tests. However, the Python libraries we use only support two-tailed tests. One common approach to obtain the one-tailed values from a two-tailed test is to half the corresponding p-value. However, this is only appropriate for symmetrical distributions. Since there is no reason to expect symmetrical distributions for our data, we present the results based on the two-tailed tests for all hypotheses. As a result, they are more conservative but we leave the option open to inspect unexpected results in the opposite direction.

We use the standard significance level of $\alpha = 0.05$ for the hypothesis tests. Using multiple statistical tests on one data set increases the likelihood of rejecting the null hypothesis when it is actually true (Type 1 error) [CFY17]. This requires proper adjustment of the significance level. Therefore, we employ Bonferroni correction [Abd07] to control for the familywise error rate. There exists no uniform interpretation of what constitutes a family ranging from every experiment in a life-time to every single test being a family [Ros96]. Therefore, we view all tests that involve the same variable in expectation as one family and as a result adjust the significance level for the nine hypothesis tests of RQ1-RQ4 to $\alpha^\star = 0.05/9 = 0.0056$. Similarly, in RQ5 we use the same test to analyze the difference of understandability assessment of two code snippets and therefore use $\alpha' = 0.05/2 = 0.025$

for RQ5. Furthermore, if additional pairwise comparisons are necessary as a result of a significant ANOVA or Kruskal-Wallis test this level is adjusted accordingly. Bonferroni adjustment is recognized as a rather conservative method potentially missing significant results [CFY17]. We apply the more conservative approaches for our analysis because we use an online survey instead of a controlled environment for the experiment. Therefore, we have little control over the participants but compensate with a more strict analysis approach. Additionally, we utilize two non-validated measurement instruments which warrants a precise analysis.

For that reason, we analyze the internal consistency of our expectation and motivation scales through Cronbach's Alpha[8]. This tests the relatedness of the single items of the scale by calculating pairwise correlations between the items. As a result, Cronbach's Alpha provides an evaluation of whether the individual items measure the same latent construct. In general, a value above 0.7 is considered acceptable, while 0.7-0.8 is respectable, 0.8-0.9 good, 0.9-0.94 excellent, and above that, there might be unnecessary redundancies [Tab17].

[8]We use the implementation of the pingouin library: https://pingouin-stats.org/generated/pingouin.cronbach_alpha.html

Chapter 5

Results

In this chapter, we present the results of our experiment and the survey it was embedded in. We cover descriptive statistics about the participant sample and show the overall outcome of the experiment.

82 people completed the survey. One participant is excluded as part of our internal review of timing statistics since they only spend seven seconds reading, understanding, and judging the code snippet. We believe that seven seconds is unreasonably low to complete this task and to provide an informed assessment about the understandability of the code snippet. Therefore, the sample for our analysis consists of 81 participants.

5.1 Descriptive Statistics

From the 81 participants, 68 identify as male, 7 as female, 1 as non-binary, and 5 preferred to not disclose this information. In terms of age the average is 32 years old with a standard deviation (SD) of 10.73. As their main occupation 43 (53%) work as IT professionals, 31 (38%) stated to be students, 6 (7%) are researchers, and 1 (1%) said they are an engineer outside of software development. For self-estimated experience on a 10-point scale, the average is 6.90 (SD = 1.95). In comparison to a 10 year expert, participants rate themselves on a 5-point scale on average as 2.37 (SD = 1.12). In regards to Java as an OO-programming language, the participants rate themselves as 4.01 (SD = 0.83) on average on a 5-point scale. Furthermore, we ask our subjects to rate the frequency of understanding code being part of their job on a 5-point scale from never (1) to always (5). The average is 3.95 (SD = 0.86) with option 4 (often) being chosen 43 times. Applying the same scale participants rate their frequency of using static analysis tools for understandability as a 2.16 (SD = 1.28) on average with option 1 (never) being selected most often namely 33 times.

5.1.1 Survey Data

Regarding the survey, it took participants on average 12:13 minutes (SD = 6:12 minutes) to finish the survey from which on average 3:07 minutes (SD = 2:24 minutes) was spent on the task of judging the code snippet. In terms of the code snippets, 35 participants saw the easy code snippet while 46 saw the hard code snippet. They rate the code snippets on a 10-point scale where 1 is easy to understand and 10 is hard to understand. The easy code snippet received an average rating of 2.40 (SD = 2.14) while the hard code snippet was rated as 5.87 (SD = 2.38) on average. Additionally, participants used more time to inspect the hard code snippet with an average of 3:56 minutes (SD = 2:21 Minutes) than to look at the easy code snippet with an average of 2:04 minutes (SD = 1:27 minutes).

At the end of the survey, we ask participants to imagine working with a static analytic tool displaying an understandability metric and let them answer statements regarding their motivation and emotions on a 5-point Likert scale from strongly disagree (1) to strongly agree (5). The participants say, that a bad understandability score would motivate them to improve their code with an average of 3.96 (SD = 0.73). Furthermore, a good score would lead to them not considering the code for refactoring with an average score of 3.16 (SD = 1.09). This indicates a wider spread, which is showcased by 36 participants choosing option 4 or 5 while 25 choose options 1 or 2. In terms of emotions, participants mostly agree, that bad understandability scores would feel bad with an average of 3.86 (SD = 0.85). The same is seen for good understandability scores and feeling satisfied with oneself with an average of 4.06 (SD = 0.66). Lastly, we ask the participants whether they think understandability metrics are accurate. The results show an average of 3.04 (SD = 0.77) where 43 participants choose option 3 neither agree nor disagree and almost equal amounts choose option 4 (18 times) and option 2 (17 times).

5.1.2 Treatment Groups

The participants were randomly split into the three treatment groups in rates of 31 participants being placed into the control group (C), 23 participants got suggested that the code snippet will be easy to understand (M_3), and 27 participants got suggested that the code snippet will be hard to understand (M_8).

These groups can be further split up into whether they saw the easy code snippet or the hard code snippet. From the control group, 17 participants saw the easy code snippet (C_easy) and 14 participants looked at the hard code snippet (C_hard). In M_3 12 participants saw the easy code snippet (M_3_easy) and 11 the hard code snippet

Group	Pre-Task					Task (Code Score)		Post-Task				
	Expec	Motiv	V	A	D	Easy	Hard	Expec	Motiv	V	A	D
C	3.95	3.68	3.61	2.35	3.23	3.12	5.86	4.10	3.38	3.42	2.45	3.06
M_3	3.97	3.78	3.48	2.48	3.09	1.92	5.27	4.13	3.70	3.61	2.39	3.13
M_8	3.91	3.81	3.85	2.56	3.48	1.33	6.19	3.93	3.40	3.22	2.41	3.52

Table 5.1: Experiment results in terms of averages in the sequence of the experiment design and split up into the three treatment groups.

(M_3_hard). The last group, M_8, is split up into 6 participants inspecting the easy code snippet (M_8_easy) and 21 participants looking at the hard code snippet (M_8_hard).

We present the overall results of our experiment in terms of averages in Table 5.1. The results are split into the three treatment groups and displayed in order of the research design. Regarding the values, the expectation_sum and motivation_sum are divided by the number of statements of the corresponding scales. Therefore, they range from 1 to 5. Likewise, valence (V), arousal (A) as well as dominance (D) range from 1 to 5. Only the code scores are on a 10-point scale. The individual values and differences are described in more detail and used for testing our hypotheses in the section below.

5.2 Hypothesis Testing

We follow the procedure described in Section 4.8 to analyze our five research questions. The necessary information for the results of the hypothesis tests is reported according to the guidelines by Harris [Har08].

5.2.1 Research Question RQ1

In RQ1 we analyze differences in the expectations depending on the treatment group. Therefore, we look at the pre-task expectation_sum of the three groups. As presented in Table 5.1 they follow our hypothesis H_{011} in terms of their averages in $M_3 = 3.97 > C = 3.95 > M_8 = 3.91$. However, to inspect whether these are significant we employ the one-way ANOVA test. The assumption of normality for the residuals is fulfilled as assessed by the Shapiro-Wilk test ($p = 0.115$) and the inspection of the corresponding histogram and Q-Q plot. Similarly, homogeneity of variance is confirmed through Levene's test ($P = 0.899$). The analysis of the one-way ANOVA test is not statistically significant

F(2,78) = 0.072, p = 0.930, η^2 = 0.002, ω^2 = 0.023 which is an estimated Cohen's d of 0.086 indicating a very small effect.

Answer to RQ1: Since the expectations between the three treatments do not significantly differ (p = 0.930), we do not reject the null hypothesis H_{001}. Therefore, presenting an understandability metric value has no significant impact on the expectations about understanding the corresponding code snippet.

5.2.2 Research Question RQ2

For RQ2 we evaluate the correlation between expectations and motivation in our sample. Correspondingly, we inspect if there exists a significant relationship between the pre-task_expectation_sum and the pre-task_motivation_sum. While the motivation_sum displays a normal distribution as assessed by the Shapiro-Wilk test (p = 0.351), the expectation_sum does not. The Shapiro-Wilk test for the expectation_sum is significant (p = 0.028) and we therefore reject the assumption of normal distribution. This result is confirmed through a non-bell-shaped histogram and a Q-Q plot with numerous data points outside the reference diagonal line. Furthermore, applying logarithmic and square root data transformations to the expectation_sum does not improve normality as assessed by the Shapiro-Wilk test (logarithmic p = 0.005, square root p = 0.018). Therefore, we apply Kendall's τ to evaluate the relationship. The alpha level $\alpha^\star = 0.0056$ is used. Analysis of the data shows a significant positive correlation between pre-task expectation_sum and pre-task motivation_sum, τ(81) = 0.253 (95% CI [0.110, 0.385])[1], p = 0.002. Converting the τ statistic into Cohen's d results in an estimate of d = 0.839 which indicates a large effect.

These results demonstrate a strong relationship, which warrants a preciser evaluation. Therefore, we analyze a possible linear relationship of expectations predicting motivation. The relationship is showcased in Figure 5.1. In addition to the linearity shown in this scatter plot, the data meet the assumptions for linear regression in terms of a mean for the residuals near zero and a normal distribution for the residuals determined by the Shapiro-Wilk test (p = 0.579) as well as by the corresponding histogram and Q-Q plot. Furthermore, the homogeneity of variance of the residuals is fulfilled as assessed by the Breusch-Pagan test (p = 0.516), the Goldfeld-Quandt test (p = 0.250), and the scatter plot of the residuals. Additionally, no significant autocorrelation of the error terms is found as evaluated through the Durbin-Watson test (d = 1.799). The simple linear regression reveals, that expectations explain a small but significant proportion of the

[1]Confidence intervals are calculated using the implementation of XiangwenWang: https://github.com/XiangwenWang/correlation

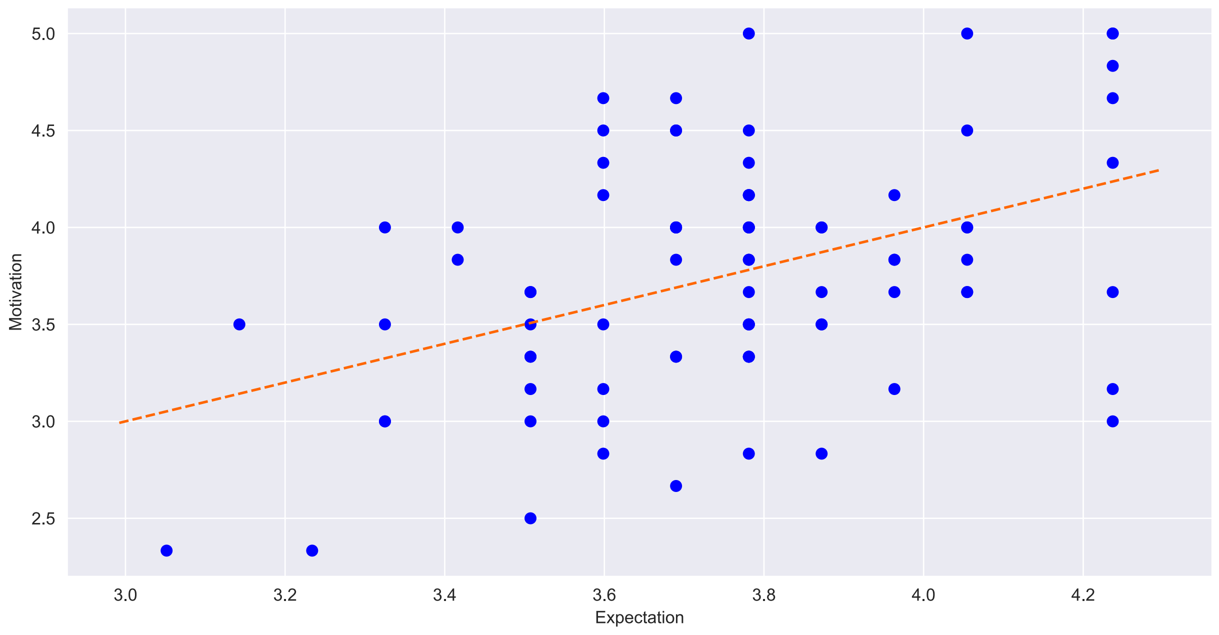


Figure 5.1: Scatter plot of the relationship between expectation and motivation with the corresponding regression line.

variance in motivation, $R^2 = 0.183$, adjusted $R^2 = 0.173$, F(1,79) = 17.73, p < 0.001. The relationship is expressed by the formula: $motivation = 1.957 + 0.456 * expectation$. The intercept 1.957 (97,5% CI [1.098, 2.816]) has a standard error of 0.432, t-statistic of 4.535, and p-value < 0.001. Similarly, the slope of 0.456 (97,5% CI [0.240, 0.671]) is significant through a standard error of 0.108, t-statistic of 4.210, and p-value < 0.001. The decently large F-statistic with p < 0.001 showcases the significance of the relationship while the relatively large t-statistics in combination with negligible p-values demonstrates the significance of the parameters. As a result of the intercept and slope values, we would expect the motivation to be on average 1.957 if the expectation would be 0, and for every increase of 1 in expectation, the motivation increases by 0.456. Additionally, a R^2 value of 0.183 suggests, that expectation can explain 18.3% of the variance in motivation.

Answer to RQ2: Expectation and motivation express a significant correlation with each other (p = 0.002) which shows a large effect ($\tau = 0.253$). More specifically, expectation explains 18.3% of the variance in motivation through a significant (p < 0.001) linear relationship expressed by: $motivation = 1.957 + 0.456 * expectation$. Therefore, we reject the null hypothesis H_{002} in favor of H_{012}. This indicates, that success expectations of understanding a code snippet significantly improve motivation in regards to engaging with the code snippet.

Affect	Shapiro-Wilk	Kendall's Tau			
Dimension	p	τ	p	95% CI	Cohen's d
Valence	<0,001	0.238	0.011	[0.095, 0.372]	0.785
Arousal	<0,001	-0.006	0.945	[-0.153, 0.141]	-0.020
Dominance	<0,001	0.137	0.134	[-0.010, 0.275]	0.437

Table 5.2: Correlation results between the pre-task expectations and the three affect dimensions valence, arousal, and dominance.

5.2.3 Research Question RQ3

To answer RQ3, we analyze the correlations between the pre-task_expectation_sum and the affect dimensions valence, arousal, and dominance. The results are presented in Table 5.2. All three dimensions fail the Shapiro-Wilk test and are therefore considered to be not normally distributed. As a result, we use Kendall's τ to measure the correlation between the pre-task expectation_sum and the three affect dimensions. Comparing the outcomes to our alpha level $\alpha^\star = 0.0056$ shows no significant result for any of the three correlations. The correlation between pre-task_expectation_sum and pre-task_valence comes close to being significant with $\tau(81) = 0.238$ (95% CI [0.095, 0.372]), p = 0.011 and estimated Cohen's d = 0.785 indicating a medium effect.

Answer to RQ3: There exists no significantly correlation between the expectations and the three affect dimensions valence (p = 0.011), arousal (p = 0.945), and dominance (p = 0.134). Therefore, we do not reject the null hypotheses H_{003}, H_{103}, and H_{203}. This means, that expectations about understanding a code snippet have no significant impact on affective states.

5.2.4 Research Question RQ4

In regards to RQ4, we inspect how differences in expectations about understanding the code snippet from pre-task to perception about how participants' understood the code snippet post-task influence the pre-task to post-task differences in motivation and affective states. Therefore, we analyze correlations between the expectation_difference and motivation_difference as well as valence_difference, arousal_difference, and dominance_difference. Overall the perception about understanding the code snippet after the task is greater than the expectations beforehand with an average difference of 0.11 (SD = 0.78). For all differences, the values can range from -4 to 4. For expectation_difference, the minimum value is -2 and the maximum 1.60. In contrast, the motivation lowered after the task with an average difference of -0.28 (SD = 0.63) as well as a minimum value of -2.33 and a maximum value of 1.17. In terms of the affect

Correlation	Shapiro-Wilk	Kendall's Tau			
Variable	p	τ	p	95% CI	Cohen's d
Motivation	0.005	0.258	0.001	[0.116, 0.390]	0.857
Valence	<0,001	0.394	<0.001	[0.262, 0.511]	1.422
Arousal	<0,001	-0.059	0.505	[-0.203, 0.089]	0.185
Dominance	<0,001	0.172	0.054	[0.026, 0.311]	0.555

Table 5.3: Correlation results between the difference of expectations pre-task and the perception after the task with the differences in motivation and the three affect dimensions valence, arousal, and dominance.

dimensions valence is lower afterwards with an average difference of -0.25 (SD = 0.93). For arousal, almost no difference exists with an average of -0.04 (SD = 0.83). The same is true for dominance with an average of -0.04 (SD = 0.73). Furthermore, there exists a clear separation between participants that see the easy code snippet having an expectation_difference average of 0.59 (SD = 0.51) and the participants that see the hard code snippet with an average of -0.26 (SD = 0.74). This is mirrored across the other variables, except for arousal where participants are slightly more aroused after seeing the hard code snippet with an average of 0.02 (SD = 0.86) in contrast to seeing the easy code snippet with an average of -0.01 (SD = 0.80).

Results of the correlations between expectation_difference and motivation_difference as well as valence_difference, arousal_difference, and dominance_difference are presented in Table 5.3. Since neither expectation_difference is normally distributed as assessed by Shapiro-Wilk (p =0.045) nor any of the four correlation variables, we use Kendall's τ to evaluate the correlations. Two significant results emerge when we compare them to the alpha level $\alpha^{\star} = 0.0056$. Expectation_difference is significantly positively correlated with the motivation_difference $\tau(81) = 0.258$ (95% CI [0.116, 0.390]), p = 0.001 and estimated Cohen's d = 0.857 indicating a large effect. Furthermore, there exists an even stronger significant positive correlation between expectation_difference and valence_difference $\tau(81) = 0.395$ (95% CI [0.262, 0.511]), p < 0.001 and estimated Cohen's d = 1.422 indicating a very large effect.

Given these results, we more closely analyze the differences between our treatment groups. The comparisons between our groups depending on the initial scenario and the code snippet participants' saw are displayed in Table 5.4. The first row after the treatment groups presents the number of participants for each group. In terms of expectation_difference the group that got suggested they will see a hard code snippet but ultimately saw the easy code snippet (M_8_easy) showcases the largest positive change with an average of 0.77 (SD = 0.53). Regarding motivation_difference the only positive change can be seen in M_3_easy who inspected the easy code snippet

Treatment	**#**	**Pre-Task to Post-Task Differences**				
Group		**Expectation**	**Motivation**	**Valence**	**Arousal**	**Dominance**
C_easy	17	0.64	-0.06	0.18	-0.06	-0.12
C_hard	14	-0.43	-0.61	-0.64	0.29	-0.21
M_3_easy	12	0.45	0.07	0.50	-0.17	0.17
M_3_hard	11	-0.16	-0.26	-0.27	0.00	-0.09
M_8_easy	6	0.77	-0.25	-0.50	-0.17	0.67
M_8_hard	21	-0.20	-0.47	-0.67	-0.14	-0.14

Table 5.4: Pre-task to post-task differences in expectation, motivation, valence, arousal, and dominance for the six treatment groups based on initial scenario and code snippet seen.

they expected to see with an average difference of 0.07 (SD = 0.64) while the largest demotivation change is showcased by C_hard with an average of -0.47 (SD = 0.73). The biggest positive change in valence is demonstrated by M_3_easy with an average of 0.50 (SD = 0.67) and the most significant negative change is displayed by C_hard with an average of -0.64 (SD = 0.93) and M_8_hard with an average of -0.67 (SD = 0.86). In contrast, the most notable result for the other two affect dimensions is seen in M_8_easy with an average increase of dominance of 0.67 (SD = 0.82).

Answer to RQ4: Differences in expectations before seeing the code snippet and the perception afterwards significantly correlate with differences in motivation (p = 0.001) with a large effect (τ = 0.258) and with differences in valence ($p < 0.001$) with a very large effect (τ = 0.394). As a result, we do not reject the null hypotheses H_{204} and H_{304} but do reject H_{004} and H_{104} in favor of the corresponding alternative hypotheses. Additionally, these differences heavily depend on the difficulty of the code snippet seen with either the easy code snippet or the hard code snippet (expectation difference of 0.59 and -0.26, motivation difference of -0.05 and -0.46, valence difference of 0.17 and -0.57). Therefore, developers who are more confident that they understood the code snippet after inspecting it than expecting to understand it beforehand are also more motivated and happier afterwards.

5.2.5 Research Question RQ5

For RQ5, we analyze the differences in subjective understandability assessment of the participants for both code snippets depending on the treatment group. The results are presented as boxplots of the three treatment groups for the easy code snippet in Figure 5.3 and for the hard code snippet in Figure 5.2. They showcase the influence

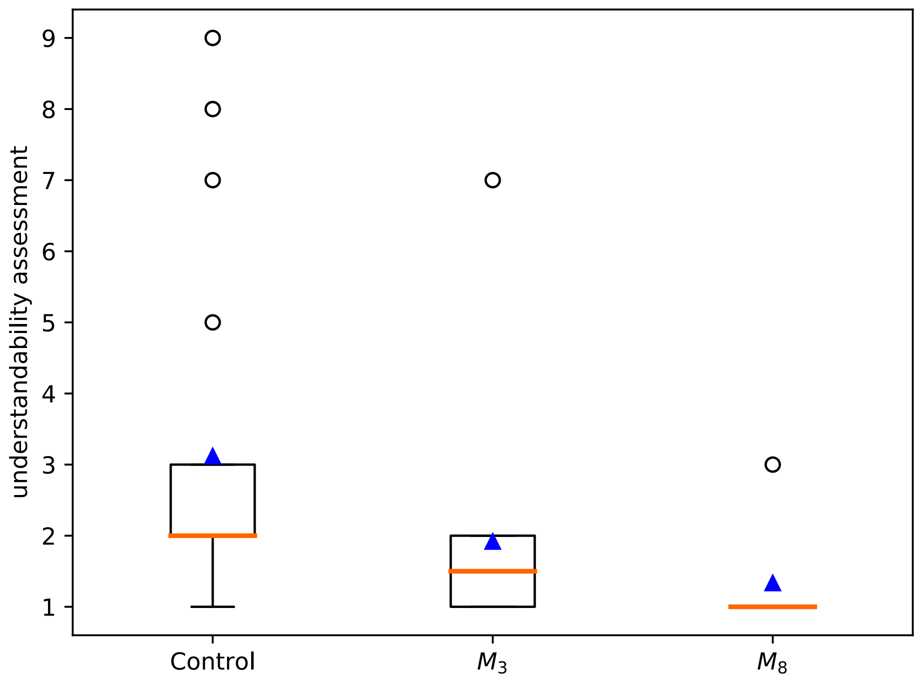


Figure 5.2: Boxplots of understandability assessment of the easy code snippet depending on the treatment group indicating the median (orange line), and the mean (blue triangle).

of outliers in the case of the easy code snippet and the similarities of the groups in the case of the hard code snippet. For the easy code snippet, the participants judge its understandability with an average score of 3.12 (SD = 2.55) for the control group, 1.92 (Sd = 1.68) for M_3, and 1.33 (SD = 0.82) for M_8. On the other side, the average score for the hard code snippet is 5.86 (SD = 2.51) for the control group, 5.27 (SD = 2.10) for M_3, and 6.19 (SD = 2.48) for M_8. To analyze whether these differences are significant, we employ the Kruskal-Wallis test since the data does not fulfill the assumptions for one-way ANOVA in terms of normality of residuals as assessed by the Shapiro-Wilk test (easy code snippet $p < 0.001$, hard code snippet $p = 0.016$) as well as by inspecting the histograms and Q-Q plots.

We use the alpha level $\alpha' = 0.025$ for these tests. For the easy code snippet, the Kruskal-Wallis test shows no significant difference in code understandability assessment between the three groups, $H(2) = 6.303$, $p = 0.043$, $\epsilon^2 = 0.185$, $\eta^2 = 0.134$ which is an estimated Cohen's d of 0.788 indicating a medium effect. Similarly, the three groups demonstrate

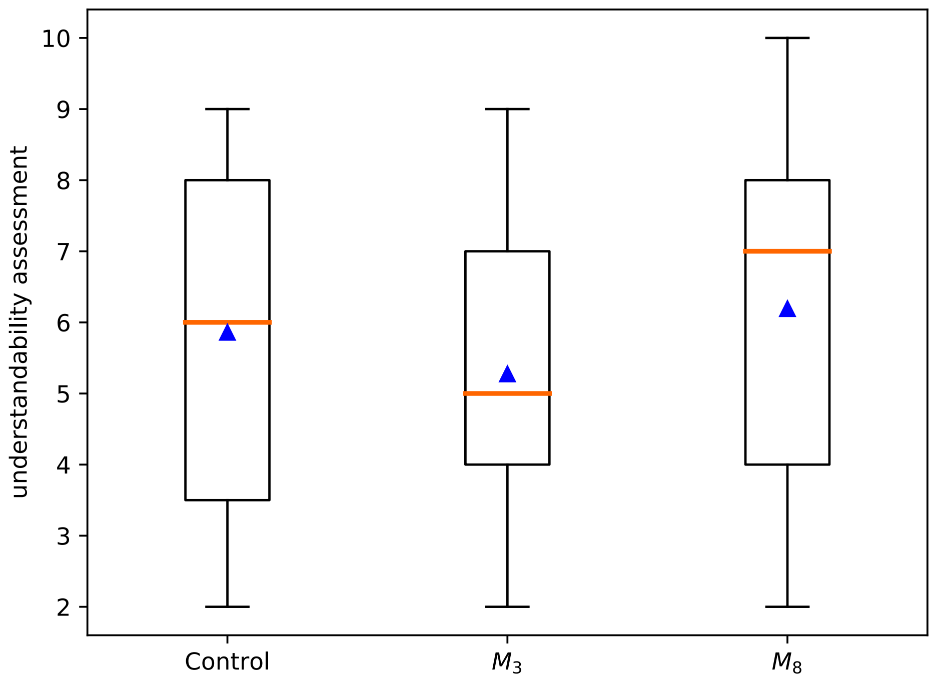


Figure 5.3: Boxplots of understandability assessment of the hard code snippet depending on the treatment group indicating the median (orange line), and the mean (blue triangle).

no significant difference for the hard code snippet, H(2) = 1.362, p = 0.506, $\epsilon^2 = 0.030$, $\eta^2 = 0.015$ which is an estimated Cohen's d of 0.245 indicating a small effect.

In order to better compare these results with the work of Preikschat [Pre20], we additionally provide the results of the Mann-Whitney-U test for the difference between M_3 and M_8. According to the Bonferroni correction, we would need to divide the α' level by an additional 2 for the secondary tests, but they are insignificant nonetheless. For the easy code snippet, the Mann-Whitney-U test shows no significant difference, U(12, 6) = 26, p = .152, $\eta^2 = 0.049$ which is an estimated Cohen's d of 0.453 indicating a small effect. The same holds true for the hard code snippet with U(11, 21) = 86.5, p = .127, $\eta^2 = 0.041$ which is an estimated Cohen's d of 0.415 indicating a small effect.

Answer to RQ5: The three treatment groups do not significantly differ in their subjective understandability assessment of neither the easy code snippet ($p = 0.043$) nor the hard code snippet ($p = 0.506$). Additionally, excluding the control group shows the same result (easy code snippet $p = 0.152$, hard code snippet $p = 0.127$). Therefore, we do not reject the null hypotheses H_{005} and H_{105}. As a result, suggesting different understandability metrics to developers before seeing a code snippet does not significantly impact their subjective assessment of the code snippet.

5.2.6 Internal Consistency of Expectation and Motivation Scales

We measure the internal consistency of our two non-validated measurement instruments through Cronbach's Alpha. The pre-task expectation scale, which is presented in Figure 4.2, has an alpha of 0.857 (95% CI [0.801, 0.901]). Similarly, the post-task expectation scale demonstrates an alpha of 0.893 (95% CI [0.851, 0.926]).

The motivation scales are displayed in Figure 4.3. In this case, the pre-task motivation scale has an alpha of 0.810 (95% CI [0.737, 0.867]). In addition, the post-task motivation scale exhibits an alpha of 0.793 (95% CI [0.715, 0.856]).

These results show good internal consistency for all four scales. Even considering the confidence internals, no scale would fall under the acceptable level of alpha = 0.7.

Chapter 6

Discussion

In this chapter, we outline and interpret the results of our experiment. Furthermore, we describe limitations regarding the applicability of our results and provide implications for the practice of software development.

6.1 Findings

When we ask the participants about their thoughts in regards to understandability metrics influencing their motivation and emotions, they report clear and strong impacts. With an average of 3.96 out of 5 participants belief, that a bad understandability score would motivate them to improve their code. On the other side, the handling of code with a good understandability score is not as unambiguous with participants disagreeing about whether such code should be ignored for refactorings or not. This might be a result of the use of refactoring as the indicator for motivation since software developers often learn that any code might be improved by a refactoring in the right situation. Furthermore, participants widely agree on the affective impact different code understandability scores have where good scores lead to satisfying feelings and bad scores lead to bad feelings. However, these opinions of our participants are not supported by the results of the experiment.

We split the participants into three groups with different scenarios of suggested code understandability scores and measure the concepts of expectations, motivation, and affective states. The groups do not differ significantly in any of these three categories. Whether we suggest a good understandability score in terms of 3 out of 10 or a bad score in terms of 8 out of 10 appears to have no effect on expectations, motivation nor the affective states. For motivation in regards to understanding and improving the code snippet, both groups are nearly identical. Furthermore, with respect to affective states, the group to which we suggested a hard to understand code snippet (M_8) appears to be happier, more excited, and feel more dominant than the other two groups. However, we ask about their affective state in regards to understanding the code snippet in this instance and not about how the understandability score made them feel. Additionally, we do not measure the initial affective state of our groups at the start of the experiment and can therefore not exclude the possibility that the groups differ as a result of dissimilar initial affective states. Still, we see the polar opposite of what participants described in the survey. Interestingly, even the control group feels happier and more in control about understanding the code snippet than the group to which we imply an upcoming simple code snippet (M_3). Apart from initial differences, one possible explanation might be, that participants with a suggested hard code snippet are more excited and feel challenged to show off their knowledge in understanding the code snippet.

For the interpretation of the results with regard to our research questions, we use $\alpha^\star = 0.0056$ as the significance level for RQ1-RQ4 and $\alpha' = 0.025$ for RQ5 as a result of the Bonferroni correction we utilize in our analysis. Additionally, all measurements range from 1 to 5 except for the understandability assessment of the code snippet which is rated on a 10-point scale.

Interpretation of RQ1 We thought, that implying different code understandability scores would impact expectations about understanding the corresponding code snippet. It would make sense for developers to be more sure about comprehending a suggested simple code snippet than a suggested difficult code snippet. However, this turns out to be not true for our experiment. The expectations we measure through five statements of typical comprehension tasks result in almost the same expectation_sum for all three groups (C = 3.95, M_3 = 3.97, M_8 = 3.91). In fact, these differences are not significant at all with a p-value of 0.93 and a negligible effect size. Therefore, we are not able to reject our null hypothesis H_{001} in this case.

We believe this is for the most part a result of our rather simple scenario. In the beginning, we describe the upcoming code snippet as being between 20-30 lines long and testing a String for a criterion. Since expectations are based on previous experiences in similar situations people tend to bring information to mind that is consistent with their prediction [ORZ96]. In this case, we assume that participants have experiences with tasks using similar descriptions and were able to solve them. This is probable for our sample because participants are experienced in general programming indicated by a 6.90 average on a 10-point scale and a 4.01 average out of 5 in regards to Java programming. As a result, participants rely more on their past experiences than on the metric value we suggest. We received feedback from a few participants who confirmed this idea. They told us that they were certain to be able to understand the code snippet with that description regardless of what the understandability metric said.

Furthermore, we might have not promoted the understandability metric enough to create an impact. The participants do not hold any strong opinions about the accuracy of understandability metrics with the majority of 43 choosing the neutral option in the corresponding survey question. Therefore, it could have required more effort to convince them of the correctness of the presented metric value. In his experiment, Preikschat [Pre20] provides a more elaborate story of a machine learning algorithm creating the understandability score. Furthermore, he shows the accuracy of the metric by presenting example code snippets with corresponding understandability scores. This is an option for the future to further cement the correctness of the presented understandability score. However, it would increase the required time for the survey and impact the subjective understandability assessment of the code snippet. If participants are presented with an example having an understandability score it would be used for comparison when the participants have to rate their code snippet.

Interpretation of RQ2 For the relationship between expectations and motivation, we expected to see a positive correlation. This is confirmed in the hypothesis test showcasing a significant positive correlation with $p = 0.002$ and $\tau = 0.253$ (95% CI [0.110, 0.385]) indicating a large effect. Considering the confidence interval on the low end, the results demonstrate a small effect while on the high end they would be very large. Since we employ a Bonferroni correction and an in general more strict analysis procedure, we can be sure with a high probability that these results are not random and relevant for our sample. Furthermore, we demonstrate the linearity of this relationship through linear regression where expectations explain 18.3% of the variance in motivation with $p < 0.001$. As a result of this regression, we would expect motivation to increase by 0.456 for every increase of 1 in expectation. Therefore, we accept the alternative hypothesis H_{012} in this case.

These results are in line with multiple motivation theories predicting the impact of expectations on motivation. Both in the expectancy theory by Vroom [Vro64] and the expectancy-value theory by Eccles et al. [EAF+83] expectancies about the success regarding a goal directly influence the motivation of achieving that goal. Therefore, if someone beliefs in their ability to accomplish a goal then they are motivated to engage in actions that bring them closer to reaching their goal. This has been shown in multiple previous studies, for example in the context of problem-solving [ELI+14] and learning to program [LLY10]. Likewise, our results show, that this relationship applies to expectations about understanding a code snippet and motivation in terms of engaging with the code snippet. However, we only partially inspect the second part of these motivation theories which Eccles et al. call value. Value describes the belief in the attractiveness or benefit of an action in achieving a goal. With the very narrow goal of understanding one specific code snippet, there does not exist much room for different evaluations of activities. Furthermore, any results would be hard to generalize since there is a clear distinction between an arbitrary code snippet in an experiment or survey and a developer's own code in terms of value. One option to further investigate value would be to present different activities one could engage with to understand the code snippet and measure the goal congruency [TF14]. In this work, we focus on aspects of intrinsic motivation, which represents one part of value, to show that success expectations about understanding code lead to increases in expected enjoyment from engaging with that code. As a result of this link, we can predict enhanced performance, persistence, and creativity for developers that expect to understand a code snippet [RD00].

Interpretation of RQ3 Similar to motivation, we believed that expectations could have an effect on affective states in terms of the three measured affect dimensions in valence, arousal, and dominance. The results show no significant correlation for expectation with any of the three dimensions (valence $p = 0.011$, arousal $p = 0.945$, dominance $p = 0.134$). Therefore, we do not reject the corresponding null hypotheses H_{003}, H_{103}, and H_{203}. For valence the correlation comes very close to being significant with $\tau = 0.238$ (95% CI [0.095, 0.372]). This represents a similar effect size and confidence interval like the correlation between expectations and motivation. Currently, there is an open debate about the meaningfulness and problems of p-values and alpha levels in hypothesis testing [Cum13]. Simply using more participants or even repeating the same experiment with a different sample has a good chance to suddenly produce significant results. Having almost the same effect size and confidence interval for the correlation between expectations and motivation and the correlation between expectations and valence indicates that the correlation between expectations and valence has a considerable likelihood to be significant when replicating the experiment [Cum13]. Additionally, the at least medium effect size gives credence to the potential relevance of this correlation.

In the end, our analysis reveals no significant correlations between expectations and affective states. Generally, optimistic expectations produce positive affect [ORZ96] which is why it might be interesting to further explore this link. Research shows the influence of positive affect on developers in terms of performance, motivation, productivity, code quality, and self-image [GFWA17; GFWA18; GWA14a; GWA14b]. In all these categories happy developers perform better. As a result, it is worth to study this effect in future work to see whether one can induce positive affect through success expectations. In their study, Graziotin, Wang, and Abrahamsson [GWA14a] found a positive correlation between valence and dominance with self-assessed productivity (sPR) which explains 38% of the variance in sPR. Similarly, our results reveal the most potential for valence and to some degree dominance to be positively related to expectations. On the other side, as with Graziotin, Wang, and Abrahamsson's study, the affect dimension of arousal, which is the intensity of emotional activation, appears to be a non-factor. Arousal in general was unimportant in our experiment since it was very similar for all groups and did not change much pre-task to post-task. This is interesting because valence and arousal are the two main dimensions of core affect theory [Rus03] as well as the pleasure-arousal-dominance model [RM77] which is the basis for SAM.

We believe that the simple scenario we use might not be enough to create a significant affect response. The lower arousal values across the board indicate that no strong affect was felt by the participants. Measurable changes in affect are typically a result of an event with an attributed affect which can lead to an emotional episode [Rus03]. Especially when people can attribute their affect change to an object, corresponding cognitive processes of emotional response are activated. The same is displayed in the theory of the impact of affects on programming performance by Graziotin, Wang, and Abrahamsson [GWA15a], where events are the starting point that trigger affects and as a result influence performance. Being told, that an arbitrary code snippet you are not familiar with will be hard or easy to understand is perhaps not sufficient to trigger an affective response. In practice, statistical analysis tools provide an evaluation for a developer's own code which typically encompasses the whole project. Therefore, developers see whether their code fulfills the quality requirements oftentimes through a color schema from red to green which could induce stronger affect responses. As a result, it might be interesting to observe this effect in a real practical environment to investigate whether static analysis tools trigger events that influence affective states.

Interpretation of RQ4 In terms of variance between pre-task to post-task measures, we expected to see multiple differences. As a result of using three different treatment groups in combination with two code snippets of distinct difficulty, we believed that there would be significant differences in the perception of participants that got suggested one difficulty of code snippet but saw the opposite code snippet. Our results reveal a significant positive correlation with a large effect between the difference of expectations before seeing the code snippet and perception of understanding the code snippet afterwards with pre-task to post-task motivation difference ($p = 0.001$, $\tau = 0.258$). We measure an even stronger effect for the correlation between the expectation difference and the valence difference with $p < 0.001$ and $\tau = 0.394$ indicating a very large effect. Similar to previous results, arousal is truly insignificant ($p = 0.505$) while dominance shows some potential ($p = 0.054$). Therefore, we reject the null hypotheses H_{004} and H_{104} in favor of the corresponding alternative hypotheses but do not reject the null hypotheses H_{204} and H_{304}.

Inspecting the pre-task to post-task differences demonstrates that motivation is generally lower after seeing the code snippet (avg = -0.28) as well as valence (avg = -0.25) while arousal and dominance remain the same (avg for both = -0.04). Differences heavily rely on the code snippet seen, where the easy code snippet increases expectations (avg = 0.59), slightly decreases motivation (avg = -0.05) and increases valence (avg = 0.17). Inspecting the hard code snippet significantly decreases all three on average (expectation -0.26, motivation -0.46, valence -0.57). For the different groups, we thought that the group to which we imply a difficult code snippet but ultimately sees the easy code snippet would be much happier while the opposite would be true for the mirrored group M_3_hard. However, the results show, that M_8_easy was not happier (avg = -0.50) and M_3_hard was only slightly unhappy (avg = -0.27) but still way happier than all other groups that saw the hard code snippet.

These results further strengthen the positive relationship between expectations and motivation. We did not investigate the causal link between the two. However, it seems reasonable to argue in accordance with results for RQ1 that developers experiencing that they understand a code snippet much better than they expected increases motivation to engage with the code snippet. In the same vein, being positively surprised by one's ability to understand a code snippet correlates with being happier after inspecting the code snippet. We see a very large effect for this relationship which is in accordance with expectancy research [ORZ96] as well as expectation-confirmation theory [Bha01]. Both explain that expectancies that are exceeded and expectancies that remain unfulfilled have an increased effect on affective states.

Furthermore, the effect of different code snippets is a result in and of itself. Even though we did not analyze this effect with statistical methods it is clear, that harder to understand code lowers motivation, valence, and dominance. This is the one case where arousal actually slightly increases for the hard code snippet (avg = 0.02) but not for the easy code snippet (avg = -0.11). Participants probably feel either more stressed or challenged by the hard code snippet which increased their arousal. However, the decrease in motivation and valence is more important since this confirms the outcomes of bad code quality. In their large scale survey, Graziotin et al. [GFWA17] found developers to report external qualities as reasons for negative affect four times as much as internal causes. These external causes are dominated by bad code quality which leads to lower motivation and lowers cognitive performance. Our results speak to the potential truth of the more exploratory claims of Graziotin et al. which warrants additional research investigating the link between bad code quality and decreased motivation as well as unhappiness. As discussed already decreases in motivation and affect can have an effect on people for longer amounts of time and beyond the current task influencing persistence, general well-being, up to work withdrawal [GFWA17; RD00].

Lastly, the unexpected differences between the groups have to be considered with the lower amount of participants for each group in mind. M_8_easy only consists of 6 participants while M_3_hard consists of 11 people. Therefore, these results are not as representative as the rest. Nevertheless, we are surprised by the lack of increased valence in M_8_easy. Maybe they expected to inspect a challenging puzzle of a code and were disappointed after seeing the easy code snippet. Interestingly, this group reported a large increase in dominance (avg = 0.67) which could mean that they feel more in control of the situation after being sure they understand the code snippet. On the other side, we thought that people in M_3_hard might be frustrated seeing a more difficult code snippet than implied in the beginning. The difference turns out to be small which could indicate that participants in this group perceive the hard code snippet to not be as difficult as the other groups as a result of the initial scenario telling them it will be simple. This is supported by the lowest pre-task to post-task expectation difference afterwards (avg = -0.16) for this group in the category of inspecting the hard code snippet. Interestingly, the control group reports the biggest overall negative difference for motivation (avg = -0.61) and dominance (avg = -0.21) as well as the most increase in arousal (avg = 0.29) for the group that saw the hard code snippet (C_hard). They did not have the added challenge of comparing themselves with a provided understandability score and therefore might question the purpose of the experiment after already having inspected the code snippet. As a result, they might feel annoyed or frustrated with having to answer the same set of questions again in addition to seeing a hard to understand code snippet. In the end, our results show that expecting simple code and seeing an easy code snippet increases motivation and valence while seeing a hard code snippet decreases motivation and valence. However, for the other case where participants expect to see a difficult code snippet both scenarios of either looking at an easy code snippet or a hard code snippet decreases motivation and valence.

Interpretation of RQ5 We expected to observe significant differences in the subjective understandability assessment based on the three initial treatment groups. The analysis shows no significant difference for both code snippets. In the case of the easy code snippet, the understandability score differs between the three groups (C = 3.12, M_3 = 1.68, M_8 = 1.33), but not significantly with p = 0.043 and $\eta^2 = 0.134$ indicating a possible medium effect. We could employ the same reasoning as we did in interpreting RQ3 for a potential significant p-value in a different sample for the easy code snippet. However, in this case, one group consists of only six participants which makes it unreasonable to predict the outcomes for larger or different samples. For the hard code snippet, we see the same result with an average assessment difference between the groups (C = 5.86, M_3 = 5.27, M_8 = 6.19) and no significance (p = 0.506, $\eta^2 = 0.030$). Excluding the control group and only analyzing M_3 and M_8 produces the unchanging insignificant results with p = 0.152 for the easy code snippet and p = 0.127 for the hard code snippet with small effects for both. Therefore, we do not reject the corresponding null hypotheses H_{005} and H_{105} for this research question.

In accordance with the anchoring effect, we would expect the participants to rely on the provided understandability value and choose scores more closely to it. We can see slight differences for the hard code snippet with M_3 rating it on average as a 5.27 and M_8 rating it as a 6.19. Therefore both groups lean a little towards the understandability score presented to them. Interestingly, for the easy code snippet, M_8 even has the lowest average score among all three groups with 1.33. However, these results showcase that the participants were not significantly influenced in their code assessment as a result of the understandability metric presented in the beginning.

Preikschat [Pre20] discovered a significant difference in subjective understandability assessment in his experiment using a similar set-up with two groups who got suggested different difficulties of upcoming code snippets. His participants appear to be highly influenced by the provided understandability metric which results in a significant difference with a p-value < 0.001 and a large effect (Cohen's d = 1.376). Similarly, Jørgensen and Sjøberg [JS04] showed, that providing an initial customer estimation value for the effort of a software project significantly influences the effort estimation of both students and professionals. Therefore, it is quite surprising to not be able to replicate these results with both of our code snippets. Especially, since anchoring effects are thought of as being one of the most persuasive and robust cognitive effects [FB11; MES04]. The effect has been shown to work with clearly uninformative values, with experienced judges, highly cognitive people, and over longer periods of time.

In his work, Preikschat promoted the presented understandability metric to a greater extent than we did. Furthermore, he showed the metric directly next to the code snippets while participants tried to understand them. In contrast, we only depict the metric in the scenario description and the code instruction on the page before participants see the code. Therefore, the metric was more present in the mind of Preikschat's participants. Still, the anchoring effect is shown to last up to a week which means that 20 seconds between seeing it in the code instructions and then looking at the code should not impact the effect. Furnham and Boo [FB11] report mixed results for the use of implausible anchors where they sometimes work more effectively but other times they are less effective. In our case, the latter appears to be true since five out of the six people selected the lowest number in 1 as the understandability score for the easy code snippet even though they were told it will be an 8 out of 10. We inspected, whether programming experience was another mitigating factor for this group, but this group has an average of 6.5 out of 10 in general programming experience and 4.2 out of 5 in Java experience in contrast to the whole sample with an average of 6.90 and 4.01 respectively. It seems like when the provided understandability score does not fit at all to the code snippet, developers are able to ignore this anchor and judge the code independently. Additionally, one participant in the opposite group of M_3_easy choose 7 for the understandability score which severely skews the results of this group. We received feedback from two participants that they might have misunderstood the scale to have said that the score of 10 would be very easy to understand. Since the survey is anonymous we can not be sure that the participant with a 7 is one of them, but it is very likely since they represent a lonely outlier. However, even excluding this participant would not change the results. For the hard code snippet, both anchors of 3 and 8 appear to be plausible scores since the code snippet was rated as a 5.86 on average by the control group. We are not sure, why the anchors only worked very slightly in this case. Perhaps a more difficult code snippet would cause clearer results.

The most important factors in interpreting these results are the small and uneven groups. For the easy code snippet, we have M_3_easy with 12 participants and M_8_easy with 6. A similar difference emerges for the hard code snippet with M_3_hard having 11 participants and M_8_hard consisting of 21. This severely limits the power of our ranked-based hypothesis test.

We propose another possibility for the results. Similar to Boot et al.'s discovery, that expectations of participants can predict the outcomes of video game interventions that claim to improve cognition [BSSS13], we believe that expectations could predict anchoring effects. According to explanatory theories of the anchoring effect, anchors lead to selective accessibility of anchor-consistent information which influences the final judgment of a subject [FB11; MES04]. This accessibility of anchor-consistent information could be measured by expectations in regards to the upcoming judgment. As a result, different anchors create different expectations about the object to be judged which could explain the difference in outcomes. In their literature reviews both Furnham and Boo [FB11] as well as Mussweiler, Englich, and Strack [MES04] do not mention the influence of expectations at all which reveals a possibility for further research. This approach would explain our results. Since we did not measure any difference in expectations as a result of the anchors, participants were not significantly influenced by it in their judgment.

6.2 Limitations

In Section 4.6.1 we already discussed our efforts to mitigate potential threats to validity through our research design. Therefore, this section describes the remaining limitations for our experiment which should be considered when interpreting the results.

6.2.1 Participant Sample

As always, more participants would increase the reliability of our results. Especially considering the combination of three treatment groups with two code snippets which creates small samples for the resulting six groups. Therefore, the comparisons between these groups for research question RQ4 and RQ5 should be considered with the necessary carefulness. They might look different for a bigger sample or a different population.

Overall, our sample showcases a wide variety of developers. In regards to age, participants are on average 32 years old but with a standard deviation of 10.73 years. Similarly, we have a good mix of 53% IT professionals, 38% students, and 7% researchers. In terms of gender, only seven participants identify as female and one as non-binary which makes it hard to generalize for these populations if differences exist. Research shows some differences between genders and comprehending code which could result in different understandability scores [SS14]. Furthermore, our participants are generally experienced programmers as indicated by an average score of 6.90 out of 10 on the scale of general programming. Therefore, we can not make statements about how inexperienced developers would perform in the experiment. In past experiments of code comprehension, researchers present non-linear differences for programming experience [AWF18]. While professionals with varying levels of experience exhibit only slight differences, there are significant differences between novices and professionals. In addition, the interpretation of affective states can differ substantially in other cultures or languages [Rus91] which should be considered when generalizing beyond English speaking developers.

In the end, we believe that our sample of 81 developers with a mix of students and professionals provides a good representation of the general developer population.

6.2.2 Use of an Online Survey

Using an online survey instead of a controlled experiment has strengths and weaknesses. On the one hand, we do not have control of the environment the participants are in. They might get distracted by other people, noise, or what else is happening on their screen. Additionally, participants might not read everything and answer with the most due diligence, because they do not feel observed. Furthermore, we can not answer questions or help with problems that occur during the experiment. On the other hand, this environment more closely represents the real world.

This trade-off between control and generalizability is typical for any experiment [Sto05]. In the software engineering community, there is no consensus on the importance of internal versus external validity and when to focus on which one [SSA15]. We choose generalizability since it is more interesting, in our opinion, to inspect constructs that are closer to reality. Furthermore, these days online surveys are easier to administer and provide the potential for more participants.

6.2.3 Measurement Instruments

We use two unvalidated scales in order to measure expectations and motivation pre-task as well as post-task. This is a result of not finding any applicable validated scales that fit our experiment setting. Especially our requirement of administrating the scales two times, before and after a task turned out to be challenging. Most scales, we identified in the literature deal with broader topics of expectations and motivation, not single tasks. Therefore, we constructed our own scales based on validated ones from the literature. Analysis of the Cronbach alpha values shows good internal consistency for all four of our scales. However, to actually validate these scales many more participants would be necessary with a dedicated experiment evaluating construct and discriminant validity, internal consistency, and retest accuracy. Especially, whether our scales actually measure the intended constructs of expectation and motivation would be important to know. Furthermore, the two scales are quite similar in the activities they describe. We believe, that "I think I could fix a bug in the code snippet" and "I would feel pleased fixing a bug in the code snippet" should yield different answers. However, we can not exclude the possibility that participants interpret them as being similar as a result of using the same activity in the sentence. In addition, since we use a similar version of these scales pre-task and post-task which only differentiate by the tense used, participants could be inclined to provide the same answers they gave before. Our results show that this is generally not true, because we see significant differences in answers between pre-task to post-task.

Different methods exist to measure motivation [TF14] as well as affective states [MR09]. Researchers suggest to use multiple instruments to compare them to each other if possible [KDL13; MR09]. We only use self-report measurements because they are the simplest to administer in an online survey. In addition, other instruments either rely on observing participants directly or are harder to interpret. For example time measurements, which are often employed to measure motivation, are difficult to interpret since we can never be sure that differences can be attributed to motivation instead of other factors such as tiredness [TF14]. Furthermore, we limit the instruments we use since we do not want to increase the required time for the survey and potentially annoy our participants. After all, the intention of the survey was described as judging the understandability of a code snippet and not filling out many questionnaires about one's feelings. Therefore, different measurement instruments could lead to divergent results.

Furthermore, the use of other code snippets could certainly produce different results. Our code snippets achieve the intended effect of being perceived as easier to understand and harder to understand respectively. An even more difficult to understand code snippet for the hard code snippet could reveal more differences than we saw. However, our code snippets are rather small and therefore easier to understand in general. With only 20 to 30 lines of code and the topic of String manipulation they represent an easy challenge for developers. In practice, projects and methods are way bigger and require domain knowledge to understand. Additionally, working with one's own code probably influences motivation and affect response differently as a result of understandability scores then working with an example in an online survey. Therefore, we are cautious to generalize the results beyond the limited scope of small and independent code snippets.

As mentioned before, we received feedback from two participants believing they might have misused the understandability scale. More specifically, they thought that a 10 on the scale from 1 to 10 would indicate an easy to understand code. However, in reality, it is the opposite where 1 is easy to understand and 10 is hard to understand. We explicitly describe this scale in the scenario description as well as directly in the question where participants have to rate the understandability of the code snippet. Still, it appears that for some people their internal perception of how the scale should look like overrides the actual description or they did not read the description. Since we use an anonymous survey, there is no way to reliably exclude these participants. Furthermore, we do not know if more than two participants had this problem. We believe, that our sample of 81 participants alleviates these problems for the most part. However, the understandability score is most important for research question RQ5 where we split our participants into six groups for comparison which increases the impact individual participants have on the results. Accordingly, these influences should be considered for the results of RQ5.

6.2.4 Confounding Factors

The research of placebos and nocebos has a long history of studying personality traits that impact these effects [KKWB20]. Evidence suggests that traits such as optimism, empathy, extraversion, ego resilience, altruism, forwardness, and low hostility relate to greater placebo response [DBC14]. Researchers acknowledge that this response is likely a combination of personality and the context of the treatment [GKH+07]. Similar research is conducted in the area of software engineering, but the amount is small and the evidence weak or inconclusive [CSM+11]. In their experiment, Wyrich, Graziotin, and Wagner [WGW19] found conscientious to be a predictor of worse performance in coding challenges. Furthermore, Preikschat [Pre20] observed anxiety to correlate with the deviation from the understandability metric. We do not measure personality traits for our participants. As discussed before, we are very happy with the scope and required time of our survey. As a result, differences in personality traits between the groups could impact their expectations, motivation, and affect reaction. The demographic data including programming experience we collect shows balanced values between the three treatment groups. However, we can not exclude that other factors we do not control for might impact participants' performance in the experiment.

For affective states, we do not measure an initial value at the start of the survey. Therefore, participants in the three groups could differ in their starting valence, arousal, and dominance. This can affect their performance in the experiment in regards to comprehending the code snippet [GFWA17; GFWA18; GWA14b; WGW19]. Furthermore, variation in the affective states would impact our results in terms of comparing the different groups in research question RQ4. One option would be to administer SPANE [DWT+09b] at the beginning of the survey to gather the overall status of positive and negative affect of the participants. However, as before we were inclined to not include more instruments in our survey.

Similar arguments can be used for considering the motivation of participants as a confounding factor. People that volunteer to participate in a survey are generally more motivated than the whole population [WRH+12]. In addition, we use an incentive of a 5€ donation which could influence motivation. However, research shows little evidence for negative effects of survey incentives and instead incentive respondents might actually produce better data quality [CSW15].

Another factor for the assessment of the code snippets might be diverse understandings of what constitutes comprehensible code between the participants. We internally discussed providing a definition of understandability at the beginning of the survey. However, since there exists neither an agreement in the literature (see Section 2.1.1) nor does it seem realistic that developers all share the same view on understandability we decided against it. Therefore, it might be possible that participants in one group share similar views on what constitutes understandable code, while participants in other groups differ. Furthermore, developers from another population could interpret the code snippets completely differently as a result of their view on understandable code.

6.2.5 Reliability of Results

In regards to research question RQ2, where we evaluate the relationship between expectations and motivation, it is important to note, that we only have high values for both corresponding scales. We use the original sums of the expectation and motivation scales without dividing them by the number of statements for this argumentation because it is easier to follow with these values. As a result, we measure an average of 19.72 out of 25 for expectation and only seven participants report a value below 16 as well as an average of 22.53 out of 30 for motivation and only seven participants below 18. For this reason, our results are only reliable for higher values. Even though the correlation and linear regression would predict increases in motivation as a result of increases in expectation for small values, the sample data does not provide corresponding measurements.

The same is true for our other research questions where expectations are involved, namely all expect for RQ5. For research question RQ4 the difference values for pre-task expectation to post-task perception only range from 8 to -10 from a potential of 20 to -20 and the difference values for motivation range from 7 to -14 in light of potential values from 25 to -25. As a result, the predictions of our correlations and linear regression are less reliable for values outside these ranges.

6.3 Implications

As it stands, our results show that understandability metrics do not induce placebo effects or contribute to a cognitive bias. At least for small code snippets in combination with decently experienced developers they should not impact their expectations about understanding the code. Therefore, no direct link to influencing motivation or affective states, which are known to impact performance, exists as a result of presenting different understandability metrics. Even though static analysis tools oftentimes utilize unvalidated metrics, this should not pose a threat to the performance of developers under the discussed limitations.

Independently of understandability metrics we see a clear relationship between expectations and motivation where expectations influence motivation. It is important to ensure for developers, that they believe in their ability to complete any task they are presented with. Even for more challenging assignments, it seems valuable to present them in a way that the programmer feels confident in their success. As a result, one can expect increased motivation in solving the task which improves performance, persistence, creativity, self-esteem, and general well-being. Similar effects could be true for positive affect as a consequence of success expectations, but this link needs further research. In the end, we encourage anyone assigning tasks to ask about as well as track expectations of the assigned person and help them with developing expectations of completing the task.

Furthermore, differences between expectation and perception in regards to understanding code impact motivation and affective states. Therefore, the expectancy-confirmation theory holds true for the context of comprehending code snippets. While it is beneficial to induce success expectations, it is similarly important to manage them in a way where these expectations are confirmed in the end. Otherwise, intensified negative effects of demotivation and negative affect could occur. On the other side, lower expectations in the beginning can result in enhanced motivation and positive affect if they are exceeded. A balancing act is needed where expectations are positive but remain reasonable. Additionally, this result showcases another reason for static analysis tools to improve their false positive rate. When warnings of static analysis tools elicit expectations about the code which are then not fulfilled actually looking at the code snippet, developers' motivation and affective state diminishes. As a result, they might ignore warnings in the future or get annoyed by them, since their expectations about the usefulness of warnings change through these experiences.

In regards to the anchoring effect, we observe no such effect for both of our code snippets. Here, it would be interesting to further evaluate in which situations anchoring is mitigated. We offer explanations in terms of an implausible anchor in combination with experience as well as the potential of expectations predicting anchoring effects. In their current form, our results reveal no anchoring effect as a result of a presented understandability metric. However, these results should be applied cautiously while taking the limitations of small sample sizes into account.

For future research, we believe that incorporating more expectation measurements in experiment designs could provide interesting information. While outcome measures oftentimes explain what happened expectations could support insights into why it happened. Researchers frequently believe that a certain treatment would have an expected effect on their participants and then go on to measuring the outcome. However, it might be beneficial to ask the participants about the expected effect and measure if this effect actually occurs. As a result, expectations should be considered as a confounding factor in experiments where treatments could induce different expectations in different treatment groups.

Chapter 7

Conclusion

The comprehension of program code is a highly cognitive task and takes up most of the working time of software developers. As a result, the use of static analysis tools has become stable in many software projects in order to improve code towards understandability. Especially for maintenance tasks, it is important to preserve an understandable codebase since developers have to comprehend the code first before incorporating changes. However, most of the used metrics in static analysis tools are not empirically validated.

Therefore, we investigated the consequences of these unvalidated metrics on the cognitive processes of software developers. We reported on an experiment evaluating the effects of presenting different understandability metric values to programmers. 81 developers consisting of students and IT professionals participated in our online survey that incorporated a code snippet assessment task with questionnaires before and after. Both times, these questionnaires asked participants about their expectations, motivation, and affective states with regard to understanding the code snippet. In an initial scenario description, we presented different understandability values for the upcoming code snippet depending on the treatment group. Participants were randomly assigned to one of three treatment groups: a control group, a group to which we imply a simple upcoming code snippet, and a group to which we imply a difficult code snippet. Additionally, we used two code snippets with contrasting difficulty to analyze the reactions of participants. Furthermore, we asked participants how they think understandability metrics would impact their motivation and affective state.

Our results show clear differences between the personal opinion of participants and their performance in the experiment. While participants think that they would be motivated by a bad understandability score and satisfied by a good score, the results do not support this thesis. Instead, our findings show no significant difference for expectations, motivation, and affective state as a result of presenting different understandability scores. However, we discover a strong positive linear relationship between expectations of understanding code and motivation to engage with that code where expectations explain 18.3% of the variance of motivation. With regard to a relationship between expectations and affective states, we do not find significant results. Inspecting the difference between expectations of understanding the code snippet before seeing it and the perception of understanding it afterwards reveals multiple results. For one, differences in expectation to perception demonstrate a strong positive correlation with differences of motivation from pre-task to post-task. Therefore, developers who are positively surprised by the easiness of understanding the code snippet are also more motivated to further engage with that code snippet. We find an even stronger positive correlation between differences in expectation and differences in valence. This means, positively surprised developers are additionally a lot happier after inspecting the code snippet. Lastly, we investigate to what extent the understandability values presented at the start of the experiment influence the subjective understandability assessment of the code snippet. Contrasting to prior work, our results show no significant impact on the assessment of the developers for both code snippets.

In the end, these results showcase a surprisingly insignificant and minimal impact of presenting different understandability values to developers. For the combination of experienced programmers and smaller code snippets, understandability metrics do not influence expectations about understanding the code nor the assessment of the code. On the other side, our findings exemplify the importance of believing in one's ability of understanding code. In both scenarios, before seeing the code snippet and after inspecting the code snippet expectations strongly impact the motivation and in the latter part additionally the happiness of developers. Therefore, we encourage anyone assigning programming tasks to manage success expectations in a way to establish them beforehand and ensure that they are fulfilled afterwards. In addition, static analysis tools should improve their false positive rate in order to mitigate expectancy disconfirmation events that decrease motivation and happiness of programmers which impacts their engagement with the corresponding code in the future. Lastly, our results highlight the benefit of specifically measuring expectations in different treatment groups to explain outcomes which should be considered in future experiments as a potential confounding factor.

Future Work

To combat the limitation of having a majority of experienced developers in our sample, we plan to replicate the experiment with a different sample of only software engineering students. As a result, we can investigate potential differences based on experience level. Additionally, combining both samples could reveal new results. Furthermore, replicating the experiment with a more difficult scenario and code snippets would be interesting to see the corresponding effect on the results.

Regarding expectancy, we believe future research can benefit from incorporating expectation measurements in order to investigate treatment effects. Furthermore, we propose to analyze whether anchoring effects can be predicted by the expectations of people.

For motivation, the intrinsic motivation inventory [DR20] offers more categories of motivation than we used in our experiment which could reveal further insights. Especially, if the influence of understandability metrics would be observed in a practical setting with developers working on their own code and live interacting with static analysis tools using metrics.

This approach could additionally uncover more realistic effects of metrics on affective states. Another method would be to investigate distinct emotions that arise as a result of different expectations and specifically expectancy disconfirmation events. Pekrun et al. [PVMS16] developed and validated a scale for epistemic emotions that emerge based on knowledge-generating tasks which consist of the emotions surprised, curious, excited, confused, anxious, frustrated, and bored. This could be applied to the context of developers trying to understand program code.

Furthermore, our results revealed vast differences in perception, motivation, and valence based on the difficulty of the inspected code snippet which warrants further experimental investigation of the impact different quality code has on developers with regard to these concepts.

Chapter 7

Bibliography

[Abd07] H. Abdi. “Bonferroni and Šidák corrections for multiple comparisons.” In: *Encyclopedia of measurement and statistics* 3 (2007), pp. 103–107 (cit. on p. 54).

[AMOK19] E. A. AlOmar, M. W. Mkaouer, A. Ouni, M. Kessentini. “On the Impact of Refactoring on the Relationship between Quality Attributes and Design Metrics.” In: *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, Sept. 2019. DOI: 10.1109/esem.2019.8870177 (cit. on p. 28).

[Apa] Apache. *Code snippet retrieved from https://github.com/apache/commons-lang/blob/master/src/main/java/org/apache/commons/lang3/StringUtils.java#L3609. The code snippet was slightly changed. It is licensed under the Apache License Version 2.0.* (Cit. on p. 43).

[ASC] K. Aggarwal, Y. Singh, J. Chhabra. “An integrated measure of software maintainability.” In: *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No.02CH37318)*. IEEE. DOI: 10.1109/rams.2002.981648 (cit. on p. 5).

[AVZ15] E. Ammerlaan, W. Veninga, A. Zaidman. “Old habits die hard: Why refactoring for understandability does not give immediate benefits.” In: *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Mar. 2015. DOI: 10.1109/saner.2015.7081865 (cit. on pp. 7, 9).

[AWF18] S. Ajami, Y. Woodbridge, D. G. Feitelson. “Syntax, predicates, idioms — what really affects code complexity?” In: *Empirical Software Engineering* 24.1 (June 2018), pp. 287–328. DOI: 10.1007/s10664-018-9628-3 (cit. on pp. 6, 7, 9, 78).

[Bau15] R. F. Baumeister. “Toward a general theory of motivation: Problems, challenges, opportunities, and the big picture.” In: *Motivation and Emotion* 40.1 (Oct. 2015), pp. 1–10. DOI: 10.1007/s11031-015-9521-y (cit. on p. 17).

[BB01] B. Boehm, V. Basili. “Top 10 list [software development].” In: *Computer* 34.1 (2001), pp. 135–137. DOI: 10.1109/2.962984 (cit. on p. 1).

[BBH+08] S. Beecham, N. Baddoo, T. Hall, H. Robinson, H. Sharp. “Motivation in Software Engineering: A systematic literature review.” In: *Information and Software Technology* 50.9-10 (Aug. 2008), pp. 860–878. DOI: 10.1016/j.infsof.2007.09.004 (cit. on pp. 31, 32).

[BBL76] B. W. Boehm, J. R. Brown, M. Lipow. “Quantitative evaluation of software quality.” In: *Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press. 1976, pp. 592–605 (cit. on pp. 5, 6).

[Ben05] F. Benedetti. “Neurobiological Mechanisms of the Placebo Effect.” In: *Journal of Neuroscience* 25.45 (Nov. 2005), pp. 10390–10402. DOI: 10.1523/jneurosci.3458-05.2005 (cit. on p. 15).

[Ber07] D. Bertram. “Likert scales.” In: *Retrieved November* 2 (2007), p. 2013 (cit. on p. 53).

[BF17] T.-M. Bynion, M. T. Feldner. “Self-assessment manikin.” In: *Encyclopedia of personality and individual differences* (2017), pp. 1–3 (cit. on p. 23).

[BG07] S. G. Barsade, D. E. Gibson. “Why Does Affect Matter in Organizations?” In: *Academy of Management Perspectives* 21.1 (Feb. 2007), pp. 36–59. DOI: 10.5465/amp.2007.24286163 (cit. on p. 32).

[BH14] K. E. Barron, C. Hulleman. “Expectancy-Value-Cost Model of Motivation.” In: Jan. 2014, pp. 503–509 (Vol. 8). DOI: 10.1016/B978-0-08-097086-8.26099-6 (cit. on p. 18).

[Bha01] A. Bhattacherjee. “Understanding Information Systems Continuance: An Expectation-Confirmation Model.” In: *MIS Quarterly* 25.3 (Sept. 2001), p. 351. DOI: 10.2307/3250921 (cit. on pp. 14, 15, 74).

[BL94] M. M. Bradley, P. J. Lang. “Measuring emotion: the self-assessment manikin and the semantic differential.” In: *Journal of behavior therapy and experimental psychiatry* 25.1 (1994), pp. 49–59 (cit. on pp. 23, 41, 53).

[BP16] J. Borstler, B. Paech. “The Role of Method Chains and Comments in Software Readability and Comprehension—An Experiment.” In: *IEEE Transactions on Software Engineering* 42.9 (Sept. 2016), pp. 886–898. DOI: 10.1109/tse.2016.2527791 (cit. on p. 6).

[Bro83] R. Brooks. "Towards a theory of the comprehension of computer programs." In: *International Journal of Man-Machine Studies* 18.6 (June 1983), pp. 543–554. DOI: 10.1016/s0020-7373(83)80031-5 (cit. on p. 12).

[BSSS13] W. R. Boot, D. J. Simons, C. Stothart, C. Stutts. "The Pervasive Problem With Placebos in Psychology." In: *Perspectives on Psychological Science* 8.4 (July 2013), pp. 445–454. DOI: 10.1177/1745691613491271 (cit. on pp. 17, 30, 77).

[BV16] A. Betella, P. F. M. J. Verschure. "The Affective Slider: A Digital Self-Assessment Scale for the Measurement of Human Emotions." In: *PLOS ONE* 11.2 (Feb. 2016). Ed. by U. S. Tran, e0148037. DOI: 10.1371/journal.pone.0148037 (cit. on p. 24).

[BW10] R. P. L. Buse, W. R. Weimer. "Learning a Metric for Code Readability." In: *IEEE Transactions on Software Engineering* 36.4 (July 2010), pp. 546–558. DOI: 10.1109/tse.2009.70 (cit. on pp. 6, 7).

[Cam18a] G. A. Campbell. "Cognitive Complexity-A new way of measuring understandability." In: *SonarSource SA* (2018) (cit. on p. 10).

[Cam18b] G. A. Campbell. "Cognitive complexity." In: *Proceedings of the 2018 International Conference on Technical Debt - TechDebt '18*. ACM Press, 2018. DOI: 10.1145/3194164.3194186 (cit. on p. 10).

[CAS+16] C. Chen, R. Alfayez, K. Srisopha, L. Shi, B. Boehm. "Evaluating Human-Assessed Software Maintainability Metrics." In: *Communications in Computer and Information Science*. Springer Singapore, 2016, pp. 120–132. DOI: 10.1007/978-981-10-3482-4_9 (cit. on p. 5).

[CAS03] J. K. Chhabra, K. Aggarwal, Y. Singh. "Code and data spatial complexity: two important software understandability measures." In: *Information and Software Technology* 45.8 (June 2003), pp. 539–546. DOI: 10.1016/s0950-5849(03)00033-8 (cit. on p. 9).

[CBS79] C. S. Carver, P. H. Blaney, M. F. Scheier. "Reassertion and giving up: The interactive role of self-directed attention and outcome expectancy." In: *Journal of Personality and Social Psychology* 37.10 (1979), pp. 1859–1870. DOI: 10.1037/0022-3514.37.10.1859 (cit. on p. 14).

[CC17] N. Corsi, L. Colloca. "Placebo and Nocebo Effects: The Advantage of Measuring Expectations and Psychological Factors." In: *Frontiers in Psychology* 8 (Mar. 2017). DOI: 10.3389/fpsyg.2017.00308 (cit. on pp. 15, 17).

[CD10] C. Croux, C. Dehon. "Influence functions of the Spearman and Kendall correlation measures." In: *Statistical Methods & Applications* 19.4 (May 2010), pp. 497–515. DOI: 10.1007/s10260-010-0142-z (cit. on p. 54).

[CF14] R. J. Casson, L. D. Farmer. "Understanding and checking the assumptions of linear regression: a primer for medical researchers." In: *Clinical & Experimental Ophthalmology* 42.6 (June 2014), pp. 590–596. DOI: 10.1111/ceo.12358 (cit. on p. 54).

[CFY17] S.-Y. Chen, Z. Feng, X. Yi. "A general introduction to adjustment for multiple comparisons." In: *Journal of Thoracic Disease* 9.6 (June 2017), pp. 1725–1729. DOI: 10.21037/jtd.2017.05.34 (cit. on pp. 54, 55).

[CLLB04] L. Colloca, L. Lopiano, M. Lanotte, F. Benedetti. "Overt versus covert treatment for pain, anxiety, and Parkinson's disease." In: *The Lancet Neurology* 3.11 (Nov. 2004), pp. 679–684. DOI: 10.1016/s1474-4422(04)00908-1 (cit. on p. 15).

[CNA+20] S. Chattopadhyay, N. Nelson, A. Au, N. Morales, C. Sanchez, R. Pandita, A. Sarma. "A Tale from the Trenches: Cognitive Biases and Software Development." In: *ACM/IEEE 42nd International Conference on Software Engineering (ICSE)*. 2020 (cit. on pp. 2, 26, 27).

[Coh88] J. Cohen. "Statistical power analysis for the behavioral sciences New York." In: *NY: Academic* (1988) (cit. on p. 54).

[CP08] J. Carifio, R. Perla. "Resolving the 50-year debate around using and misusing Likert scales." In: *Medical education* 42.12 (2008), pp. 1150–1152 (cit. on p. 53).

[CS17] C. Carbonell-Carrera, J. L. Saorın. "Geospatial Google Street View with Virtual Reality: A Motivational Approach for Spatial Training Education." In: *ISPRS International Journal of Geo-Information* 6.9 (Aug. 2017), p. 261. DOI: 10.3390/ijgi6090261 (cit. on p. 20).

[CSLB19] C. Chen, M. Shoga, B. Li, B. Boehm. "Assessing Software Understandability in Systems by Leveraging Fuzzy Method and Linguistic Analysis." In: *Procedia Computer Science* 153 (2019), pp. 17–26. DOI: 10.1016/j.procs.2019.05.051 (cit. on pp. 1, 6, 8).

[CSM+11] S. Cruz, F. da Silva, C. Monteiro, C. Santos, M. dos Santos. "Personality in software engineering: preliminary findings from a systematic literature review." In: *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*. IET, 2011. DOI: 10.1049/ic.2011.0001 (cit. on p. 80).

[CSW15] J. S. Cole, S. A. Sarraf, X. Wang. "Does use of survey incentives degrade data quality?" In: Association for Institutional Research Annual Forum. 2015 (cit. on p. 81).

[Cum13] G. Cumming. "The New Statistics." In: *Psychological Science* 25.1 (Nov. 2013), pp. 7–29. DOI: 10.1177/0956797613504966 (cit. on p. 72).

[DB00] G. J. Devilly, T. D. Borkovec. "Psychometric properties of the credibility/-expectancy questionnaire." In: *Journal of Behavior Therapy and Experimental Psychiatry* 31.2 (June 2000), pp. 73–86. DOI: 10.1016/s0005-7916(00)00012-4 (cit. on p. 17).

[DBC14] M. Darragh, R. J. Booth, N. S. Consedine. "Investigating the 'placebo personality' outside the pain paradigm." In: *Journal of Psychosomatic Research* 76.5 (May 2014), pp. 414–421. DOI: 10.1016/j.jpsychores.2014.02.011 (cit. on p. 80).

[Dec86] E. L. Deci. "Ryan. RM (1985). Intrinsic motivation and self-determination in human behavior." In: *New York and London: Plenum* (86) (cit. on pp. 18, 19).

[DK11] B. Dwyer, Y. Kim. "For Love or Money: Developing and Validating a Motivational Scale for Fantasy Football Participation." In: *Journal of Sport Management* 25.1 (Jan. 2011), pp. 70–83. DOI: 10.1123/jsm.25.1.70 (cit. on p. 20).

[DR20] E. L. Deci, R. M. Ryan. "Intrinsic motivation inventory (IMI)." In: *Retrieved from https://selfdeterminationtheory.org/intrinsic-motivation-inventory* (April of 2020) (cit. on pp. 20, 41, 87).

[DSC13] H. Ding, H. Sun, A. Chen. "Impact of Expectancy-Value and Situational Interest Motivation Specificity on Physical Education Outcomes." In: *Journal of Teaching in Physical Education* 32.3 (July 2013), pp. 253–269. DOI: 10.1123/jtpe.32.3.253 (cit. on p. 20).

[DVMK17] J. Dietrich, J. Viljaranta, J. Moeller, B. Kracke. "Situational expectancies and task values: Associations with students' effort." In: *Learning and Instruction* 47 (Feb. 2017), pp. 53–64. DOI: 10.1016/j.learninstruc.2016.10.009 (cit. on p. 20).

[DWT+09a] E. Diener, D. Wirtz, W. Tov, C. Kim-Prieto, D. Choi, S. Oishi, R. Biswas-Diener. "Scale of Positive and Negative Experience (SPANE)." In: *Ed Diener personal Website* (2009) (cit. on p. 23).

[DWT+09b] E. Diener, D. Wirtz, W. Tov, C. Kim-Prieto, D.-w. Choi, S. Oishi, R. Biswas-Diener. "New Well-being Measures: Short Scales to Assess Flourishing and Positive and Negative Feelings." In: *Social Indicators Research* 97.2 (May 2009), pp. 143–156. DOI: 10.1007/s11205-009-9493-y (cit. on pp. 23, 81).

[EA17] A. F. Ernst, C. J. Albers. "Regression assumptions in clinical psychology research practice—a systematic review of common misconceptions." In: *PeerJ* 5 (May 2017), e3323. DOI: 10.7717/peerj.3323 (cit. on p. 54).

[EAF+83] J. S. Eccles, T. F. Adler, R. Futterman, S. Goff, C. Kaczala, J. Meece, C. Midgley, J. Spence. "Achievement and achievement motivation." In: *Expectancies, values and academic behaviors* (1983), pp. 75–146 (cit. on pp. 18, 71).

[Edw00] K. Edwards. "When beggers become choosers." In: *First Monday* 5.10 (2000) (cit. on p. 6).

[Ekm92] P. Ekman. "An argument for basic emotions." In: *Cognition and Emotion* 6.3-4 (May 1992), pp. 169–200. DOI: 10.1080/02699939208411068 (cit. on p. 21).

[ELI+14] D. Eseryel, V. Law, D. Ifenthaler, X. Ge, R. Miller. "An investigation of the interrelationships between motivation, engagement, and complex problem solving in game-based learning." In: *Educational technology & society* 17.1 (2014), pp. 42–53 (cit. on pp. 31, 32, 71).

[Erl00] L. Erlikh. "Leveraging legacy system dollars for e-business." In: *IT Professional* 2.3 (2000), pp. 17–23. DOI: 10.1109/6294.846201 (cit. on p. 1).

[EW95] J. S. Eccles, A. Wigfield. "In the Mind of the Actor: The Structure of Adolescents' Achievement Task Values and Expectancy-Related Beliefs." In: *Personality and Social Psychology Bulletin* 21.3 (Mar. 1995), pp. 215–225. DOI: 10.1177/0146167295213003 (cit. on p. 18).

[FALK11] J. Feigenspan, S. Apel, J. Liebig, C. Kastner. "Exploring Software Measures to Assess Program Comprehension." In: *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, Sept. 2011. DOI: 10.1109/esem.2011.21 (cit. on pp. 7, 9).

[FB11] A. Furnham, H. C. Boo. "A literature review of the anchoring effect." In: *The Journal of Socio-Economics* 40.1 (Feb. 2011), pp. 35–42. DOI: 10.1016/j.socec.2010.10.008 (cit. on pp. 12, 13, 76, 77).

[FBM+14] T. Fritz, A. Begel, S. C. Müller, S. Yigit-Elliott, M. Züger. "Using psycho-physiological measures to assess task difficulty in software development." In: *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, 2014. DOI: 10.1145/2568225.2568266 (cit. on p. 11).

[Fea66] N. T. Feather. "Effects of prior success and failure on expectations of success and subsequent performance." In: *Journal of Personality and Social Psychology* 3.3 (1966), pp. 287–298. DOI: 10.1037/h0022965 (cit. on p. 14).

[FGS+11] A. Franca, T. Gouveia, P. Santos, C. Santana, F. da Silva. “Motivation in software engineering: a systematic review update.” In: *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*. IET, 2011. DOI: 10.1049/ic.2011.0019 (cit. on pp. 31, 32).

[FHFB05] A. J. Fairchild, S. J. Horst, S. J. Finney, K. E. Barron. “Evaluating existing and new validity evidence for the Academic Motivation Scale.” In: *Contemporary Educational Psychology* 30.3 (July 2005), pp. 331–358. DOI: 10.1016/j.cedpsych.2004.11.001 (cit. on p. 20).

[Fla] Flaticon. *Icons provided by www.flaticon.com. Icons made by Freepik (peoples, questionnaires, easy code snippet), Vectors Market (3 and 8), and prettycons (hard code snippet).* (Cit. on p. 36).

[FMAA18] S. Fakhoury, Y. Ma, V. Arnaoudova, O. Adesope. “The Effect of Poor Source Code Lexicon and Readability on Developers’ Cognitive Load.” In: *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*. 2018, pp. 286–28610 (cit. on pp. 11, 42).

[FN07] G. A. Fredricks, R. B. Nelsen. “On the relationship between Spearman’s rho and Kendall’s tau for pairs of continuous random variables.” In: *Journal of Statistical Planning and Inference* 137.7 (July 2007), pp. 2143–2150. DOI: 10.1016/j.jspi.2006.06.045 (cit. on p. 54).

[Fow18] M. Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018 (cit. on p. 5).

[FRHA19] S. Fakhoury, D. Roy, A. Hassan, V. Arnaoudova. “Improving Source Code Readability: Theory and Practice.” In: *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, May 2019. DOI: 10.1109/icpc.2019.00014 (cit. on p. 28).

[FTAS08] R. Feldt, R. Torkar, L. Angelis, M. Samuelsson. “Towards individualized software engineering.” In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering - CHASE ’08*. ACM Press, 2008. DOI: 10.1145/1370114.1370127 (cit. on p. 32).

[GFWA17] D. Graziotin, F. Fagerholm, X. Wang, P. Abrahamsson. “On the Unhappiness of Software Developers.” In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE’17*. ACM Press, 2017. DOI: 10.1145/3084226.3084242 (cit. on pp. 33, 72, 74, 81).

[GFWA18] D. Graziotin, F. Fagerholm, X. Wang, P. Abrahamsson. “What happens when software developers are (un)happy.” In: *Journal of Systems and Software* 140 (June 2018), pp. 32–47. DOI: 10.1016/j.jss.2018.02.041 (cit. on pp. 34, 72, 81).

[GKH+07] A. L. Geers, K. Kosbab, S. G. Helfer, P. E. Weiland, J. A. Wellman. "Further evidence for individual differences in placebo responding: An interactionist perspective." In: *Journal of Psychosomatic Research* 62.5 (May 2007), pp. 563–570. DOI: 10.1016/j.jpsychores.2006.12.005 (cit. on p. 80).

[Gra16] D. Graziotin. "Towards a Theory of Affect and Software Developers' Performance." In: (Jan. 2016) (cit. on pp. 23, 32).

[GVB00] F. Guay, R. J. Vallerand, C. Blanchard. "On the assessment of situational intrinsic and extrinsic motivation: The Situational Motivation Scale (SIMS)." In: *Motivation and Emotion* 24.3 (2000), pp. 175–213. DOI: 10.1023/a:1005614228250 (cit. on pp. 18, 20).

[GWA14a] D. Graziotin, X. Wang, P. Abrahamsson. "Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering." In: *Journal of Software: Evolution and Process* 27.7 (Aug. 2014), pp. 467–487. DOI: 10.1002/smr.1673 (cit. on pp. 33, 34, 51, 53, 72).

[GWA14b] D. Graziotin, X. Wang, P. Abrahamsson. "Happy software developers solve problems better: psychological measurements in empirical software engineering." In: *PeerJ* 2 (Mar. 2014), e289. DOI: 10.7717/peerj.289 (cit. on pp. 32, 33, 72, 81).

[GWA15a] D. Graziotin, X. Wang, P. Abrahamsson. "How do you feel, developer? An explanatory theory of the impact of affects on programming performance." In: *PeerJ Computer Science* 1 (Aug. 2015), e18. DOI: 10.7717/peerj-cs.18 (cit. on pp. 32, 33, 73).

[GWA15b] D. Graziotin, X. Wang, P. Abrahamsson. "Understanding the affect of developers: theoretical background and guidelines for psychoempirical software engineering." In: *Proceedings of the 7th International Workshop on Social Software Engineering - SSE 2015*. ACM Press, 2015. DOI: 10.1145/2804381.2804386 (cit. on pp. 21, 22, 24, 32, 53).

[Har08] P. Harris. *Designing and reporting experiments in psychology*. McGraw-Hill Education (UK), 2008 (cit. on p. 59).

[HBB+09] T. Hall, N. Baddoo, S. Beecham, H. Robinson, H. Sharp. "A systematic review of theory use in studies investigating the motivations of software engineers." In: *ACM Transactions on Software Engineering and Methodology* 18.3 (May 2009), pp. 1–29. DOI: 10.1145/1525880.1525883 (cit. on p. 31).

[HPMB16] E. C. S. Hayashi, J. E. G. Posada, V. R. M. L. Maike, M. C. C. Baranauskas. "Exploring new formats of the Self-Assessment Manikin in the design with children." In: *Proceedings of the 15th Brazilian Symposium on Human Factors in Computer Systems - IHC '16*. ACM Press, 2016. DOI: 10.1145/3033701.3033728 (cit. on p. 24).

[HSS+19] P. Hurst, L. Schipof-Godart, A. Szabo, J. Raglin, F. Hettinga, B. Roelands, A. Lane, A. Foad, D. Coleman, C. Beedie. "The Placebo and Nocebo effect on sports performance: A systematic review." In: *European Journal of Sport Science* (Aug. 2019), pp. 1–14. DOI: 10.1080/17461391.2019.1655098 (cit. on p. 15).

[IM19] G. Iturregui-Gallardo, J. L. Méndez-Ulrich. "Towards the Creation of a Tactile Version of the Self-Assessment Manikin (T-SAM) for the Emotional Assessment of Visually Impaired People." In: *International Journal of Disability, Development and Education* (June 2019), pp. 1–18. DOI: 10.1080/1034912x.2019.1626007 (cit. on p. 24).

[IRFW19] N. Imtiaz, A. Rahman, E. Farhana, L. Williams. "Challenges with Responding to Static Analysis Tool Alerts." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, May 2019. DOI: 10.1109/msr.2019.00049 (cit. on p. 29).

[ISO11] ISO/IEC 25010. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. 2011 (cit. on p. 5).

[Iza13] C. E. Izard. *Human emotions*. Springer Science & Business Media, 2013 (cit. on p. 21).

[Jam04] S. Jamieson. "Likert scales: how to (ab)use them." In: *Medical Education* 38.12 (Dec. 2004), pp. 1217–1218. DOI: 10.1111/j.1365-2929.2004.02012.x (cit. on p. 53).

[JCP08] A. Jedlitschka, M. Ciolkowski, D. Pfahl. "Reporting Experiments in Software Engineering." In: *Guide to Advanced Empirical Software Engineering*. Springer London, 2008, pp. 201–228. DOI: 10.1007/978-1-84800-044-5_8 (cit. on p. 35).

[JF14] A. Jbara, D. G. Feitelson. "On the effect of code regularity on comprehension." In: *Proceedings of the 22nd International Conference on Program Comprehension - ICPC 2014*. ACM Press, 2014. DOI: 10.1145/2597008.2597140 (cit. on p. 9).

[JS04] M. Jørgensen, D. I. Sjøberg. "The impact of customer expectation on software development effort estimates." In: *International Journal of Project Management* 22.4 (May 2004), pp. 317–325. DOI: 10.1016/s0263-7863(03)00085-1 (cit. on pp. 29, 30, 76).

[JSMB13] B. Johnson, Y. Song, E. Murphy-Hill, R. Bowdidge. "Why don't software developers use static analysis tools to find bugs?" In: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, May 2013. DOI: 10.1109/icse.2013.6606613 (cit. on p. 29).

[KDL+12] O. Klein, S. Doyen, C. Leys, P. A. M. de Saldanha da Gama, S. Miller, L. Questienne, A. Cleeremans. "Low Hopes, High Expectations." In: *Perspectives on Psychological Science* 7.6 (Nov. 2012), pp. 572–584. DOI: 10.1177/1745691612463704 (cit. on p. 48).

[KDL13] S. Kaplan, R. Dalal, J. Luchman. "Measurement of emotions." In: Jan. 2013, pp. 61–75 (cit. on pp. 23, 79).

[KHBG14] J. J. Kosovich, C. S. Hulleman, K. E. Barron, S. Getty. "A Practical Measure of Student Motivation." In: *The Journal of Early Adolescence* 35.5-6 (Nov. 2014), pp. 790–816. DOI: 10.1177/0272431614556890 (cit. on p. 17).

[KK81] P. R. Kleinginna, A. M. Kleinginna. "A categorized list of emotion definitions, with suggestions for a consensual definition." In: *Motivation and Emotion* 5.4 (Dec. 1981), pp. 345–379. DOI: 10.1007/bf00992553 (cit. on p. 21).

[KKWB20] A. Kern, C. Kramm, C. M. Witt, J. Barth. "The influence of personality traits on the placebo/nocebo response." In: *Journal of Psychosomatic Research* 128 (Jan. 2020), p. 109866. DOI: 10.1016/j.jpsychores.2019.109866 (cit. on p. 80).

[Kol16] H. G. Koller. "Effects of Clean Code on Understandability: An Experiment and Analysis." MA thesis. 2016 (cit. on pp. 7, 9).

[KW13] N. Kasto, J. Whalley. "Measuring the difficulty of code comprehension tasks using software metrics." In: *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136*. 2013, pp. 59–65 (cit. on pp. 7, 9).

[LBC+97] P. J. Lang, M. M. Bradley, B. N. Cuthbert, et al. "International affective picture system (IAPS): Technical manual and affective ratings." In: *NIMH Center for the Study of Emotion and Attention* 1 (1997), pp. 39–58 (cit. on pp. 23, 42).

[LBW13] F. Li, X. Bai, Y. Wang. “The Scale of Positive and Negative Experience (SPANE): Psychometric Properties and Normative Data in a Large Chinese Sample.” In: *PLoS ONE* 8.4 (Apr. 2013). Ed. by F. Krueger, e61137. DOI: 10.1371/journal.pone.0061137 (cit. on p. 23).

[LCD99] R. D. Lane, P. M.-L. Chua, R. J. Dolan. “Common effects of emotional valence, arousal and attention on neural activation during visual processing of pictures.” In: *Neuropsychologia* 37.9 (Aug. 1999), pp. 989–997. DOI: 10.1016/s0028-3932(99)00017-2 (cit. on pp. 21, 22).

[Len16] A. Lenhard W. Lenhard. *Calculation of Effect Sizes*. 2016. DOI: 10.13140/RG.2.2.17823.92329. URL: https://www.psychometrica.de/effect_size.html (cit. on p. 54).

[LF06] J. W. Lang, S. Fries. “A Revised 10-Item Version of the Achievement Motives Scale.” In: *European Journal of Psychological Assessment* 22.3 (Jan. 2006), pp. 216–224. DOI: 10.1027/1015-5759.22.3.216 (cit. on p. 20).

[LFW15] P. Lenberg, R. Feldt, L. G. Wallgren. “Behavioral software engineering: A definition and systematic literature review.” In: *Journal of Systems and Software* 107 (Sept. 2015), pp. 15–37. DOI: 10.1016/j.jss.2015.04.084 (cit. on p. 29).

[Lik32] R. Likert. “A technique for the measurement of attitudes.” In: *Archives of psychology* (1932) (cit. on p. 8).

[LL02] E. A. Locke, G. P. Latham. “Building a practically useful theory of goal setting and task motivation: A 35-year odyssey.” In: *American Psychologist* 57.9 (Sept. 2002), pp. 705–717. DOI: 10.1037/0003-066x.57.9.705 (cit. on pp. 18, 38).

[LLY10] K. M. Law, V. C. Lee, Y. Yu. “Learning motivation in e-learning facilitated computer programming courses.” In: *Computers & Education* 55.1 (Aug. 2010), pp. 218–228. DOI: 10.1016/j.compedu.2010.01.007 (cit. on pp. 32, 71).

[Lun11a] F. C. Lunenburg. “Expectancy Theory of Motivation: Motivating by Altering Expectations Sam Houston State University.” In: *International Journal Of Management, Business, And Dministration* 15.1 (2011) (cit. on p. 18).

[Lun11b] F. C. Lunenburg. “Goal-setting theory of motivation.” In: *International journal of management, business, and administration* 15.1 (2011), pp. 1–6 (cit. on p. 18).

[LWBM10] E. Y. Leng, W. Z. bte Wan Ali, R. Baki, R. Mahmud. "Stability of the Intrinsic Motivation Inventory (IMI) For the Use of Malaysian Form One Students in ICT Literacy Class." In: *EURASIA Journal of Mathematics, Science and Technology Education* 6.3 (Dec. 2010). DOI: 10.12973/ejmste/75241 (cit. on p. 20).

[MAFB07] J. J. Mao, K. Armstrong, J. T. Farrar, M. A. Bowman. "Acupuncture Expectancy Scale: Development and Preliminary Validation in China." In: *EXPLORE* 3.4 (July 2007), pp. 372–377. DOI: 10.1016/j.explore.2006.12.003 (cit. on p. 17).

[Mar01] A. J. Martin. "The Student Motivation Scale: A Tool for Measuring and Enhancing Motivation." In: *Journal of Psychologists and Counsellors in Schools* 11 (Nov. 2001), pp. 1–20. DOI: 10.1017/s1037291100004301 (cit. on p. 20).

[Mar09] R. C. Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009 (cit. on pp. 6, 7).

[MB16] E. Marshall, E. Boggis. "The statistics tutor's quick guide to commonly used statistical tests." In: *Statstutor Community Project* (2016), pp. 1–57 (cit. on p. 53).

[McC76] T. McCabe. "A Complexity Measure." In: *IEEE Transactions on Software Engineering* SE-2.4 (Dec. 1976), pp. 308–320. DOI: 10.1109/tse.1976.233837 (cit. on pp. 9, 27).

[MCF01] G. Matthews, S. E. Campbell, S. Falconer. "Assessment of Motivational States in Performance Environments." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 45.13 (Oct. 2001), pp. 906–910. DOI: 10.1177/154193120104501302 (cit. on p. 20).

[MDT89] E. McAuley, T. Duncan, V. V. Tammen. "Psychometric Properties of the Intrinsic Motivation Inventory in a Competitive Sport Setting: A Confirmatory Factor Analysis." In: *Research Quarterly for Exercise and Sport* 60.1 (Mar. 1989), pp. 48–58. DOI: 10.1080/02701367.1989.10607413 (cit. on p. 20).

[MES04] T. Mussweiler, B. Englich, F. Strack. "10 Anchoring effect." In: *Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory* (2004), p. 183 (cit. on pp. 12, 13, 76, 77).

[MF16] S. C. Müller, T. Fritz. "Using (bio)metrics to predict code quality online." In: *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*. ACM Press, 2016. DOI: 10.1145/2884781.2884803 (cit. on p. 11).

[Mit82] T. R. Mitchell. “Motivation: New Directions for Theory, Research, and Practice.” In: *Academy of Management Review* 7.1 (Jan. 1982), pp. 80–88. DOI: 10.5465/amr.1982.4285467 (cit. on p. 17).

[MKN+07] C. Mallett, M. Kawabata, P. Newcombe, A. Otero-Forero, S. Jackson. “Sport motivation scale-6 (SMS-6): A revised six-factor sport motivation scale.” In: *Psychology of Sport and Exercise* 8.5 (Sept. 2007), pp. 600–614. DOI: 10.1016/j.psychsport.2006.12.005 (cit. on p. 20).

[MML15] R. Minelli, A. Mocci, M. Lanza. “I Know What You Did Last Summer - An Investigation of How Developers Spend Their Time.” In: *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, May 2015. DOI: 10.1109/icpc.2015.12 (cit. on p. 1).

[MMP15] V. Monteiro, L. Mata, F. Peixoto. “Intrinsic Motivation Inventory: Psychometric Properties in the Context of First Language and Mathematics Learning.” In: *Psicologia: Reflexão e Crıtica* 28.3 (Sept. 2015), pp. 434–443. DOI: 10.1590/1678-7153.201528302 (cit. on p. 20).

[Moo09] A. Moors. “Theories of emotion causation: A review.” In: *Cognition & Emotion* 23.4 (June 2009), pp. 625–662. DOI: 10.1080/02699930802645739 (cit. on p. 21).

[MR09] I. B. Mauss, M. D. Robinson. “Measures of emotion: A review.” In: *Cognition & Emotion* 23.2 (Feb. 2009), pp. 209–237. DOI: 10.1080/02699930802204677 (cit. on pp. 23, 79).

[MST+18] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, P. Ralph. “Cognitive Biases in Software Engineering: A Systematic Mapping Study.” In: *IEEE Transactions on Software Engineering* (2018), pp. 1–1. DOI: 10.1109/tse.2018.2877759 (cit. on pp. 2, 26, 27).

[Muñ19] M. Muñoz Barón. *A Validation of Cognitive Complexity as a Measure of Source Code Understandability*. en. 2019. DOI: 10.18419/OPUS-10663 (cit. on pp. 7, 8).

[MV95] A. V. Mayrhauser, A. Vans. “Program comprehension during software maintenance and evolution.” In: *Computer* 28.8 (1995), pp. 44–55. DOI: 10.1109/2.402076 (cit. on p. 12).

[MWGK02] J. D. Morris, C. Woo, J. A. Geason, J. Kim. “The Power of Affect: Predicting Intention.” In: *Journal of Advertising Research* 42.3 (May 2002), pp. 7–17. DOI: 10.2501/jar-42-3-7-17 (cit. on p. 23).

[MWW20] M. Muñoz Barón, M. Wyrich, S. Wagner. “An Empirical Validation of Cognitive Complexity as a Measure of Source Code Understandability.” In: *arXiv preprint arXiv:2007.12520* (2020) (cit. on p. 10).

[MXB10] J. J. Mao, S. X. Xie, M. A. Bowman. "Uncovering the expectancy effect: the validation of the acupuncture expectancy scale." In: *Alternative therapies in health and medicine* 16 (6 2010), pp. 22–27. ISSN: 1078-6791. ppublish (cit. on p. 17).

[NAG19] M. Nilson, V. Antinyan, L. Gren. "Do Internal Software Quality Tools Measure Validated Metrics?" In: *Product-Focused Software Process Improvement*. Springer International Publishing, 2019, pp. 637–648. DOI: 10.1007/978-3-030-35333-9_50 (cit. on pp. 1, 27, 28).

[NKU+14] T. Nakagawa, Y. Kamei, H. Uwano, A. Monden, K. Matsumoto, D. M. German. "Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: a controlled experiment." In: *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*. ACM Press, 2014. DOI: 10.1145/2591062.2591098 (cit. on p. 11).

[Nor10] G. Norman. "Likert scales, levels of measurement and the "laws" of statistics." In: *Advances in Health Sciences Education* 15.5 (Feb. 2010), pp. 625–632. DOI: 10.1007/s10459-010-9222-y (cit. on p. 53).

[NS19] N. Novielli, A. Serebrenik. "Sentiment and Emotion in Software Engineering." In: *IEEE Software* 36.5 (Sept. 2019), pp. 6–23. DOI: 10.1109/ms.2019.2924013 (cit. on p. 32).

[ORZ96] J. Olson, N. Roese, M. Zanna. "Expectancies." In: (1996). Ed. by E. T. H. A. W. Kruglanski, pp. 211–238 (cit. on pp. 13, 14, 70, 72, 74).

[Pen87] N. Pennington. "Stimulus structures and mental representations in expert comprehension of computer programs." In: *Cognitive Psychology* 19.3 (July 1987), pp. 295–341. DOI: 10.1016/0010-0285(87)90007-7 (cit. on p. 12).

[Pet08] S. Petter. "Managing user expectations on software projects: Lessons from the trenches." In: *International Journal of Project Management* 26.7 (Oct. 2008), pp. 700–712. DOI: 10.1016/j.ijproman.2008.05.014 (cit. on pp. 29, 30).

[PFB08] D. D. Price, D. G. Finniss, F. Benedetti. "A Comprehensive Review of the Placebo Effect: Recent Advances and Current Thought." In: *Annual Review of Psychology* 59.1 (Jan. 2008), pp. 565–590. DOI: 10.1146/annurev.psych.59.113006.095941 (cit. on pp. 2, 15, 16).

[PHD11] D. Posnett, A. Hindle, P. Devanbu. "A simpler model of software readability." In: *Proceeding of the 8th working conference on Mining software repositories - MSR '11*. ACM Press, 2011. DOI: 10.1145/1985441.1985454 (cit. on p. 6).

[PLB18] J. Pantiuchina, M. Lanza, G. Bavota. "Improving Code: The (Mis) Perception of Quality Metrics." In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Sept. 2018. DOI: 10.1109/icsme.2018.00017 (cit. on pp. 1, 27, 28).

[Plu82] R. Plutchik. "A psychoevolutionary theory of emotions." In: *Social Science Information* 21.4-5 (July 1982), pp. 529–553. DOI: 10.1177/053901882021004003 (cit. on p. 21).

[PMK+99] D. D. Price, L. S. Milling, I. Kirsch, A. Duff, G. H. Montgomery, S. S. Nicholls. "An analysis of factors that contribute to the magnitude of placebo analgesia in an experimental paradigm." In: *Pain* 83.2 (Nov. 1999), pp. 147–156. DOI: 10.1016/s0304-3959(99)00081-0 (cit. on pp. 15, 17).

[Pre20] A. D. Preikschat. "Experimentelle Untersuchung des Placeboeffekts beim Verstehen von Quellcode." MA thesis. 2020. DOI: http://dx.doi.org/10.18419/opus-10963 (cit. on pp. 2, 25–27, 42, 52, 66, 71, 76, 80).

[PSA+20] N. Peitek, J. Siegmund, S. Apel, C. Kastner, C. Parnin, A. Bethmann, T. Leich, G. Saake, A. Brechmann. "A Look into Programmers' Heads." In: *IEEE Transactions on Software Engineering* 46.4 (Apr. 2020), pp. 442–462. DOI: 10.1109/tse.2018.2863303 (cit. on pp. 1, 11).

[PVMS16] R. Pekrun, E. Vogl, K. R. Muis, G. M. Sinatra. "Measuring emotions during epistemic activities: the Epistemically-Related Emotion Scales." In: *Cognition and Emotion* 31.6 (July 2016), pp. 1268–1276. DOI: 10.1080/02699931.2016.1204989 (cit. on p. 87).

[Ray91] D. R. Raymond. "Reading source code." In: *Proceedings of the 1991 conference of the Centre for Advanced Studies on Collaborative research*. 1991, pp. 3–16 (cit. on p. 1).

[RD00] R. M. Ryan, E. L. Deci. "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being." In: *American Psychologist* 55.1 (2000), pp. 68–78. DOI: 10.1037/0003-066x.55.1.68 (cit. on pp. 19, 72, 74).

[RM77] J. A. Russell, A. Mehrabian. "Evidence for a three-factor theory of emotions." In: *Journal of Research in Personality* 11.3 (Sept. 1977), pp. 273–294. DOI: 10.1016/0092-6566(77)90037-x (cit. on pp. 21, 73).

[RMI+17] L. Rozenkrantz, A. E. Mayo, T. Ilan, Y. Hart, L. Noy, U. Alon. "Placebo can enhance creativity." In: *PLOS ONE* 12.9 (Sept. 2017). Ed. by E. Manalo, e0182466. DOI: 10.1371/journal.pone.0182466 (cit. on p. 15).

[Ros96] W. F. Rosenberger. “Dealing with multiplicities in pharmacoepidemiologic studies.” In: *Pharmacoepidemiology and drug safety* 5.2 (1996), pp. 95–100 (cit. on p. 54).

[Rug00] S. Rugaber. In: *Annals of Software Engineering* 9.1/4 (2000), pp. 143–192. DOI: 10.1023/a:1018976708691 (cit. on p. 1).

[Rus03] J. A. Russell. “Core affect and the psychological construction of emotion.” In: *Psychological Review* 110.1 (2003), pp. 145–172. DOI: 10.1037/0033-295x.110.1.145 (cit. on pp. 20–22, 73).

[Rus09] J. A. Russell. “Emotion, core affect, and psychological construction.” In: *Cognition & Emotion* 23.7 (Nov. 2009), pp. 1259–1283. DOI: 10.1080/02699930902809375 (cit. on p. 22).

[Rus80] J. A. Russell. “A circumplex model of affect.” In: *Journal of Personality and Social Psychology* 39.6 (1980), pp. 1161–1178. DOI: 10.1037/h0077714 (cit. on p. 22).

[Rus91] J. A. Russell. “Culture and the categorization of emotions.” In: *Psychological Bulletin* 110.3 (1991), pp. 426–450. DOI: 10.1037/0033-2909.110.3.426 (cit. on pp. 21, 78).

[RW97] V. Ramalingam, S. Wiedenbeck. “An empirical study of novice program comprehension in the imperative and object-oriented styles.” In: *Papers presented at the seventh workshop on Empirical studies of programmers - ESP ’97*. ACM Press, 1997. DOI: 10.1145/266399.266411 (cit. on p. 6).

[SAE+18] C. Sadowski, E. Aftandilian, A. Eagle, L. Miller-Cushon, C. Jaspan. “Lessons from building static analysis tools at Google.” In: *Communications of the ACM* 61.4 (Mar. 2018), pp. 58–66. DOI: 10.1145/3188720 (cit. on pp. 1, 27).

[Saw09] S. S. Sawilowsky. “New Effect Size Rules of Thumb.” In: *Journal of Modern Applied Statistical Methods* 8.2 (Nov. 2009), pp. 597–599. DOI: 10.22237/jmasm/1257035100 (cit. on p. 54).

[SBS18] P. Schober, C. Boer, L. A. Schwarte. “Correlation Coefficients.” In: *Anesthesia & Analgesia* 126.5 (May 2018), pp. 1763–1768. DOI: 10.1213/ane.0000000000002864 (cit. on p. 54).

[SBV+17] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vasquez, D. Poshyvanyk, R. Oliveto. “Automatically assessing code understandability: How far are we?” In: *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Oct. 2017. DOI: 10.1109/ase.2017.8115654 (cit. on pp. 1, 7–10, 27).

[SBV+19] S. Scalabrino, G. Bavota, C. Vendome, M. Linares-Vasquez, D. Poshyvanyk, R. Oliveto. “Automatically Assessing Code Understandability.” In: *IEEE Transactions on Software Engineering* (2019), pp. 1–1. DOI: 10.1109/tse.2019.2901468 (cit. on pp. 6, 10).

[SE84] E. Soloway, K. Ehrlich. “Empirical Studies of Programming Knowledge.” In: *IEEE Transactions on Software Engineering* SE-10.5 (Sept. 1984), pp. 595–609. DOI: 10.1109/tse.1984.5010283 (cit. on p. 12).

[She88] M. Shepperd. “A critique of cyclomatic complexity as a software metric.” In: *Software Engineering Journal* 3.2 (1988), p. 30. DOI: 10.1049/sej.1988.0003 (cit. on p. 9).

[SI94] M. Shepperd, D. Ince. “A critique of three metrics.” In: *Journal of Systems and Software* 26.3 (Sept. 1994), pp. 197–210. DOI: 10.1016/0164-1212(94)90011-6 (cit. on pp. 1, 27).

[Sie16] J. Siegmund. “Program Comprehension: Past, Present, and Future.” In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, Mar. 2016. DOI: 10.1109/saner.2016.35 (cit. on pp. 7, 12).

[SK06] P. Steel, C. J. König. “Integrating Theories of Motivation.” In: *Academy of Management Review* 31.4 (Oct. 2006), pp. 889–913. DOI: 10.5465/amr.2006.22527462 (cit. on p. 17).

[SKA+14] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich, G. Saake, A. Brechmann. “Understanding understanding source code with functional magnetic resonance imaging.” In: *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, 2014. DOI: 10.1145/2568225.2568252 (cit. on p. 11).

[SKL+13] J. Siegmund, C. Kästner, J. Liebig, S. Apel, S. Hanenberg. “Measuring and modeling programming experience.” In: *Empirical Software Engineering* 19.5 (Dec. 2013), pp. 1299–1334. DOI: 10.1007/s10664-013-9286-4 (cit. on p. 38).

[SM79] B. Shneiderman, R. Mayer. “Syntactic/semantic interactions in programmer behavior: A model and experimental results.” In: *International Journal of Computer & Information Sciences* 8.3 (June 1979), pp. 219–238. DOI: 10.1007/bf00977789 (cit. on p. 12).

[SP04] S. Stewart-Williams, J. Podd. “The Placebo Effect: Dissolving the Expectancy Versus Conditioning Debate.” In: *Psychological Bulletin* 130.2 (2004), pp. 324–340. DOI: 10.1037/0033-2909.130.2.324 (cit. on pp. 2, 15, 16).

[SPB16] K. A. Schwarz, R. Pfister, C. Büchel. “Rethinking Explicit Expectations: Connecting Placebos, Social Cognition, and Contextual Perception.” In: *Trends in Cognitive Sciences* 20.6 (June 2016), pp. 469–480. DOI: 10.1016/j.tics.2016.04.001 (cit. on pp. 2, 15).

[SS14] J. Siegmund, J. Schumann. “Confounding parameters on program comprehension: a literature survey.” In: *Empirical Software Engineering* 20.4 (May 2014), pp. 1159–1192. DOI: 10.1007/s10664-014-9318-8 (cit. on pp. 46, 49, 78).

[SSA13] M. M. S. Sarwar, S. Shahzad, I. Ahmad. “Cyclomatic complexity: The nesting problem.” In: *Eighth International Conference on Digital Information Management (ICDIM 2013)*. IEEE, Sept. 2013. DOI: 10.1109/icdim.2013.6693981 (cit. on p. 9).

[SSA15] J. Siegmund, N. Siegmund, S. Apel. “Views on Internal and External Validity in Empirical Software Engineering.” In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. IEEE, May 2015. DOI: 10.1109/icse.2015.24 (cit. on p. 78).

[STB00] R. J. Sanchez, D. M. Truxillo, T. N. Bauer. “Development and examination of an expectancy-based measure of test-taking motivation.” In: *Journal of Applied Psychology* 85.5 (2000), pp. 739–750. DOI: 10.1037/0021-9010.85.5.739 (cit. on p. 17).

[Sto05] M.-A. Storey. “Theories, methods and tools in program comprehension: past, present and future.” In: *13th International Workshop on Program Comprehension (IWPC’05)*. IEEE, 2005. DOI: 10.1109/wpc.2005.38 (cit. on pp. 8, 78).

[Tab17] K. S. Taber. “The Use of Cronbach’s Alpha When Developing and Reporting Research Instruments in Science Education.” In: *Research in Science Education* 48.6 (June 2017), pp. 1273–1296. DOI: 10.1007/s11165-016-9602-2 (cit. on p. 55).

[TBG+18] Z. Turi, E. Bjørkedal, L. Gunkel, A. Antal, W. Paulus, M. Mittner. “Evidence for Cognitive Placebo and Nocebo Effects in Healthy Individuals.” In: *Scientific Reports* 8.1 (Nov. 2018). DOI: 10.1038/s41598-018-35124-w (cit. on pp. 2, 15).

[TCM+18] A. Trockman, K. Cates, M. Mozina, T. Nguyen, C. Kästner, B. Vasilescu. “" Automatically assessing code understandability" reanalyzed: combined metrics matter.” In: *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE. 2018, pp. 314–318 (cit. on p. 10).

[TCS05] H.-L. Tuan, C.-C. Chin, S.-H. Shieh. "The development of a questionnaire to measure students' motivation towards science learning." In: *International Journal of Science Education* 27.6 (Jan. 2005), pp. 639–654. DOI: 10.1080/0950069042000323737 (cit. on p. 20).

[TF14] M. Touré-Tillery, A. Fishbach. "How to Measure Motivation: A Guide for the Experimental Social Psychologist." In: *Social and Personality Psychology Compass* 8.7 (July 2014), pp. 328–341. DOI: 10.1111/spc3.12110 (cit. on pp. 19, 72, 79).

[Tho07] E. R. Thompson. "Development and Validation of an Internationally Reliable Short-Form of the Positive and Negative Affect Schedule (PANAS)." In: *Journal of Cross-Cultural Psychology* 38.2 (Mar. 2007), pp. 227–242. DOI: 10.1177/0022022106297301 (cit. on p. 23).

[Tia11] R. Tiarks. "What maintenance programmers really do: An observational study." In: *Workshop on Software Reengineering*. Citeseer. 2011, pp. 36–37 (cit. on p. 1).

[TK74] A. Tversky, D. Kahneman. "Judgment under Uncertainty: Heuristics and Biases." In: *Science* 185.4157 (Sept. 1974), pp. 1124–1131. DOI: 10.1126/science.185.4157.1124 (cit. on pp. 12, 13).

[TN14] P. Tripathy, K. Naik. *Software Evolution and Maintenance*. John Wiley & Sons, Inc., Nov. 2014. DOI: 10.1002/9781118964637 (cit. on pp. 1, 5).

[Vet17] T. R. Vetter. "Fundamentals of Research Data and Variables." In: *Anesthesia & Analgesia* 125.4 (Oct. 2017), pp. 1375–1380. DOI: 10.1213/ane.0000000000002370 (cit. on p. 54).

[VPB+92] R. J. Vallerand, L. G. Pelletier, M. R. Blais, N. M. Briere, C. Senecal, E. F. Vallieres. "The Academic Motivation Scale: A Measure of Intrinsic, Extrinsic, and Amotivation in Education." In: *Educational and Psychological Measurement* 52.4 (Dec. 1992), pp. 1003–1017. DOI: 10.1177/0013164492052004025 (cit. on p. 20).

[VPP+19] C. Vassallo, S. Panichella, F. Palomba, S. Proksch, H. C. Gall, A. Zaidman. "How developers engage with static analysis tools in different contexts." In: *Empirical Software Engineering* 25.2 (Nov. 2019), pp. 1419–1457. DOI: 10.1007/s10664-019-09750-5 (cit. on pp. 1, 27).

[Vro64] V. H. Vroom. "Work and motivation." In: (1964) (cit. on pp. 18, 71).

[VVH97] A. Von Mayrhauser, A. M. Vans, A. E. Howe. "Program understanding behaviour during enhancement of large-scale software." In: *Journal of Software Maintenance: Research and Practice* 9.5 (1997), pp. 299–327 (cit. on p. 1).

[w3r] w3resource. *Code snippet retrieved from https://www.w3resource.com/java-exercises/string/java-string-exercise-32.php under the title "Java String Exercises: Find longest Palindromic Substring within a string." The code snippet was slightly changed. It is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.* (Cit. on p. 44).

[WA15] T. D. Wager, L. Y. Atlas. “The neuroscience of placebo effects: connecting context, learning and health.” In: *Nature Reviews Neuroscience* 16.7 (June 2015), pp. 403–418. DOI: 10.1038/nrn3976 (cit. on pp. 2, 16).

[WAB09] F. Wedyan, D. Alrmuny, J. M. Bieman. “The Effectiveness of Automated Static Analysis Tools for Fault Detection and Refactoring Prediction.” In: *2009 International Conference on Software Testing Verification and Validation*. IEEE, Apr. 2009. DOI: 10.1109/icst.2009.21 (cit. on pp. 1, 27).

[Wal03] D. A. Walker. “JMASM9: Converting Kendall’s Tau For Correlational Or Meta-Analytic Analyses.” In: *Journal of Modern Applied Statistical Methods* 2.2 (Nov. 2003), pp. 525–530. DOI: 10.22237/jmasm/1067646360 (cit. on p. 54).

[WB95] E. Weissinger, D. L. Bandalos. “Development, Reliability and Validity of a Scale to Measure Intrinsic Motivation in Leisure.” In: *Journal of Leisure Research* 27.4 (Dec. 1995), pp. 379–400. DOI: 10.1080/00222216.1995.11949756 (cit. on p. 20).

[WCT88] D. Watson, L. A. Clark, A. Tellegen. “Development and validation of brief measures of positive and negative affect: the PANAS scales.” In: *Journal of personality and social psychology* 54.6 (1988), p. 1063 (cit. on pp. 21, 23).

[WGW19] M. Wyrich, D. Graziotin, S. Wagner. “A theory on individual characteristics of successful coding challenge solvers.” In: *PeerJ Computer Science* 5 (Feb. 2019), e173. DOI: 10.7717/peerj-cs.173 (cit. on pp. 33, 80, 81).

[WRH+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-29044-2 (cit. on pp. 46–48, 81).

[WT85] D. Watson, A. Tellegen. “Toward a consensual structure of mood.” In: *Psychological Bulletin* 98.2 (1985), pp. 219–235. DOI: 10.1037/0033-2909.98.2.219 (cit. on p. 21).

[XBL+18] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, S. Li. “Measuring Program Comprehension: A Large-Scale Field Study with Professionals.” In: *IEEE Transactions on Software Engineering* 44.10 (Oct. 2018), pp. 951–976. DOI: 10.1109/tse.2017.2734091 (cit. on p. 1).

[YGHM12] J. Younger, V. Gandhi, E. Hubbard, S. Mackey. “Development of the Stanford Expectations of Treatment Scale (SETS): A tool for measuring patient outcome expectancy in clinical trials.” In: *Clinical Trials: Journal of the Society for Clinical Trials* 9.6 (Nov. 2012), pp. 767–776. DOI: 10.1177/1740774512465064 (cit. on p. 17).

[ZFT+11] S. Zaniboni, F. Fraccaroli, D. M. Truxillo, M. Bertolino, T. N. Bauer. “Training valence, instrumentality, and expectancy scale (T-VIES-it).” In: *Journal of Workplace Learning* 23.2 (Feb. 2011), pp. 133–151. DOI: 10.1108/13665621111108792 (cit. on p. 17).

All links were last followed on October 26, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Böblingen, 27.10.2020, Lasse Merz

place, date, signature