Universität Stuttgart

The Influence of Personality on Software Quality

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Erica Constanze Weilemann geb. Janke

aus Neu-Ulm

Hauptberichter:	Prof. Dr. Stefan Wagner
Mitberichter:	Prof. Dr. Philipp Brune
Tag der mündlichen Prüfung:	01. April 2021

Institut für Software Engineering

Abteilung Empirisches Software Engineering

2020

Contents

Zusammenfassung

Ziel der Arbeit ist eine Untersuchung zum Zusammenhang zwischen der Persönlichkeit eines Softwareingenieurs und der Qualität der Software, die sie/er hergestellt hat, vorrangig in Bezug auf Wartbarkeit.

Die Evaluierung erfolgte sowohl thematisch durch die Untersuchung verschiedener Aspekte als auch methodisch durch die Anwendung unterschiedlicher Forschungsmethoden.

In einem ersten Schritt wurde eine systematische Literaturrecherche durchgeführt, die bereits untersuchte Zusammenhänge zwischen Persönlichkeit und Softwarequalität beschreibt und aufdeckt. Hier zeigten sich zum Beispiel methodische Diskrepanzen hinsichtlich der verwendeten Tests zur Ermittlung der Persönlichkeit betrifft, da diese ihrerseits in der Psychologie unterschiedlich bewertet und teilweise auch kritisch diskutiert werden. Darüber hinaus sind die meisten Probanden der Untersuchungen Studierende. Somit sind Schlussfolgerungen für die Wirtschaft kritisch zu hinterfragen da psychologische Studien belegen, dass sich die Persönlichkeit eines Menschen mit Eintritt ins Berufsleben stark verändert.

Deshalb wurde der Fokus der vorliegenden Arbeit auf Experten aus der Praxis gelegt. Ein Ziel war es herauszufinden, wie Experten den idealen Softwareingenieur sehen und welchen Einfluss aus ihrer Sicht Einzelne im Softwareentwicklungsteam auf die Qualität der Software haben. Hier-

zu wurden 12 Experten aus der Praxis mit unterschiedlichen Rollen im Softwareentwicklungsprozess und aus verschiedenen Branchen in einer qualitativen Studie interviewt. Bei dieser Studie verwendeten wir einen Persönlichkeitstest, dessen Reliabilität und Validität international nachgewiesen und akzeptiert ist. Als Ergebnis ließen sich für die aus dem Softwarelifecycle resultierenden Rollen eines Softwareentwicklerteams für jede einzelne Rolle ein individuelles "Wunsch-Persönlichkeitsprofil" erstellen.

Im quantitativen Teil dieser Arbeit erstellten wir mit Hilfe einer weiteren Studie ein Modell, das 17 Metriken, die mit Softwarewartbarkeit zusammenhängen, anhand der Persönlichkeit des Softwareentwicklers vorhersagt. Hierfür wurde die Bayes'sche Datenanalysenmethode herangezogen. In diesem Teil der Arbeit zeigte sich, dass es für einzelne Wartbarkeitsmetriken durchaus Unterschiede bei den verschiedenen Persönlichkeiten gibt.

Die Literaturrecherche deckte ergänzend dazu auf, dass neben Persönlichkeit auch das Geschlecht einen Einfluss auf die Effizienz oder Produktivität eines Softwareentwicklungsteams hat. Da Persönlichkeit in diesem Zusammenhang nicht ausschlaggebend war, haben wir ergänzend zu den bereits erlangten Ergebnissen eine weitere qualitative Studie mit einer Gruppe von Studierenden durchgeführt, um zu untersuchen, wie sich die Geschlechtsverteilung auf die Effizienz bei agiler Softwareentwicklung des Teams auswirkt. Als Ergebnis der Studie resultiert zum Beispiel die Empfehlung zur Besetzung eines Scrum Masters.

Fazit:

Die Ergebnisse haben eine Relevanz für Wissenschaftler, da sie Forschungslücken im Bereich "Zusammenhang zwischen Persönlichkeit und Softwarequalität" aufzeigen sowie Methodenvorschläge für die Untersuchungen zum Zusammenhang zwischen Persönlichkeit und Wartbarkeit enthalten.

Für den Praktiker in der Wirtschaft besteht die Relevanz dieser Arbeit darin, dass sie Vorschläge zur Besetzung einzelner Rollen im Softwareentwicklungsteam enthält und Aufschluss gibt über den Zusammenhang zwischen der Persönlichkeit eines Softwareentwicklers und der Wartbarkeit der Software, die sie/er hergestellt hat.

Abstract

Objective of the work is an investigation into the relationship between the personality of a software engineer and the quality of the software she/he has created, primarily in terms of maintainability.

The evaluation was carried out both thematically and methodically.

In a first step a systematic literature review was carried out, which describes and uncovers already examined connections between personality and software quality. Here, for example, methodological discrepancies were found with regard to the tests used to determine personality, since these, in turn, are evaluated differently in psychology and sometimes also discussed critically. In addition, most of the participants in the studies are students. Thus conclusions for practitioners are to be questioned critically since psychological studies prove that the personality of humans with entrance into the working life changes strongly.

For this reason, the focus of the present work was placed on experts from practice. One goal was to find out how experts see the ideal software engineer and what influence individuals in the software development team have on the quality of the software from their point of view. In a qualitative study, 12 practical experts with different roles in the software development process and from different industries were interviewed. In this study we used a personality test whose reliability and validity has been internationally proven

and accepted. As a result, an individual “desire personality profile” could be created for each individual role for the roles of a software development team resulting from the software lifecycle.

In the quantitative part of this work, we conducted another study to create a model that predicts 17 metrics related to software maintainability based on the personality of the software developer. The Bayesian data analysis method was used for this purpose. This part of the work showed that there are differences between different personalities for individual maintainability metrics.

The literature review also revealed that, in addition to personality, gender also has an influence on the efficiency or productivity of a software development team. Since personality was not a decisive factor in this context, we conducted a qualitative study with a group of students to investigate how gender distribution affects the efficiency of agile software development. One result of the study was the recommendation to fill a Scrum Master.

Conclusion:

The results are relevant for scientists, as they show research gaps in the area of “connection between personality and software quality” and contain method suggestions for the investigations of the connection between personality and maintainability.

For the business practitioner, the relevance of this work lies in the fact that it contains suggestions for filling individual roles in the software development team and provides information about the connection between the personality of a software developer and the maintainability of the software she/he has developed.

Acknowledgements

I would like to thank Prof. Dr. Stefan Wagner for his always positive, motivating nature and constructive comments. I would like to thank Prof. Dr. Philipp Brune for all his support, be it in terms of methodical, financial, time or personal possibilities. I would like to thank Prof. Dr. Dany Meyer for her personal support.

I would like to thank the companies that have participated in the data collection. And also the HNU data center for their constant support during my experiments. Thanks also to Jan-Peter Ostberg, who always helped me as a mentor at the University of Stuttgart and with whom I was able to conduct successful experiments for our studies.

Thank goes to Mitja Weilemann and Jessica Gabb for their constructive support and the very helpful discussions on methodological and ethical issues.

I would like to thank my family for all their personal support and that they have always been at my side in all ups and downs during all these years.

In particular, I would like to thank my mother Amalia Janke, who allowed me to have time off for this work and who always lovingly looked after my little son.

Thank you Mitja for strengthening and motivating me in the critical phase at the end.

To my family.

“You use software nearly every instant you’re awake. [...] And this may sound weirdly obvious, but every single one of those pieces of software was written by a programmer [...]. Sometimes it seems that the software we use just sort of sprang into existence, like grass growing on the lawn. But it didn’t. It was created by someone who wrote out – in code – a long, painstaking set of instructions telling the computer precisely what to do, step-by-step, to get a job done. [...]
Programmers are thus among the most quietly influential people on the planet. [...] The decisions they make guide our behavior.”
(Clive Thompson [1])

[1] Coders: Who They Are, What They Think And How They Are Changing Our World, Clive Thompson, Picador, 2019.

Chapter 1

Introduction

Searching for a good book about the history of software engineering turns out to be very difficult. There is one book – "The Technical and Social History of Software Engineering"[JPSG14] – but the reviews indicate it's not written comprehensively. Other books focus more on important personalities in the field of software engineering [DG12; KDG13]. The internet provides a few more pages with short summaries (e.g. [And19; Vik; Wir]) and on youtube you can find a really entertaining talk by Paolo Perrotta about the history of software engineering [Per12].

All of them do not even agree on who coined the term "software engineering". But all of them agree that in 1968 there was an important conference for software engineering – the NATO conference which took place from 7th to 11th October 1968 in Garmisch, Germany. This conference is said to have been the first "Software Engineering Conference". The intention of this conference "was to shed further light on the many current problems in software engineering, and also to discuss possible techniques, methods and developments which might lead to their solution. It was hoped that the Conference would be able to identify present necessities, shortcomings and trends and that the findings could serve as a signpost to manufacturers of computers as well as their users" [19]. Problems which were very widespread at that time were: "projects consistently failed to deliver reliably, on time and on budget" [Vik]. The term "software crisis" describes all these problems that have been occurring since the mid-60s. And based on Dijkstra's definition of the software crisis[1], the software crisis is not over yet.

Although there have been major improvements both in programming (e.g. a shift to object orientation) and process management (e.g. the use of version management, agile methods), many problems are still unsolved today or new problems are added due to the rapid technical development. This concerns especially "human aspects of software engineering" [And19]. The present work is intended to contribute to solving some of these problems which contain human factors, and to present possible solutions.

1.1. Motivation

In order to meet the requirement to deliver software on time, new process models were developed and implemented. Nowadays, a very widespread method for software development – and in the meantime also for other processes in companies – is agile (software) development. If in the past one had the picture of a developer sitting in a quiet chamber in front of his/her computer writing code, now the picture of software developers is very often people working together in teams. This development has also been studied from a psychological point of view. Researchers have investigated which personality traits are widely spread among programmers and if their personality traits are different than the average population. In 1979, Woodruff found a different distribution of personality profiles of data processing personnel than the average population [Woo79b]. In her PHD thesis, in 1983 Allison analyzed the personality profile of computer programmers in Oklahoma and found that they were more reserved, brighter, more sober,

[1] "The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem." [Dij72]

more shy and more self-sufficient than the average population [All]. Her results underline the "old" picture of a programmer. Newer studies also investigated the personality profiles of programmers. Licorish et al. investigated "personality profiles of global software developers" and found that group practitioners who communicated a high number of messages "demonstrated more openness to experience than the Other practitioners. Additionally, practitioners involved in usability-related tasks were found to be highly extroverted, and coders were most neurotic and conscientious."[LM14]. This change of personality was also observed by Varona et al. They found that extroverts among programmers are increasing and introverts are decreasing over the last 30 years [VCPR12]. Basili states that "the development of solutions is still based upon individual creativity" [BBM96]. Karimi et al. found a link between personality and programming style [KBGW16]. These and many other studies show that personal factors have an influence on software development. But what is this influence?

1.2. Problem Statement

As shown above the concept of personality has long been included in studies in the field of software engineering. That is not surprising. Personality type or personality traits were found to be one significantly influencing factor e.g. on team performance [dFS+13; YOCC17]. This work explores the influence of personality on the quality of the outcome of a software engineering process – the software.

Chapter 3 presents the results of a structured literature review that examines which personal factors of software developers – especially which personality traits – have an influence on software and also on which aspects of software. Results of this review indicate that some small studies already proved the influence of personality on various software quality attributes ([AGJ09; DM05; IMI10; OS10; PKS06]). The literature review also identified the following gaps:

- Far too few investigations have been made in the industrial environment, most study subjects were students.

- The number of studied subjects was too small to make statistically significant statements.
- The validity and reliability of the personality type tests used were questionable.

1.3. Research Goals

The aim of my research was to differ from previous investigations in the following points:

- The study subjects will be programmers from an industrial environment. AND
- I will use a personality type test which is reliable and valid. AND
- The number of study subjects will be at least 200 to be able to draw statistically significant conclusions.

1.4. Contributions

I was able to implement the first two points. One Chapter (4) which deals with data gathered for statistical evaluation shows, that it was impossible in this context to realize the last point of my research goals.

As mentioned above, software development models have changed during the last decades. Research in the field of software engineering has tried to provide assistance on how individual software development roles should be staffed with the help of personality theory to increase performance (see e.g. [AJM06; CA10a; RMSA12]). But – as the results of the literature review indicate – most of these studies either use personality tests whose reliability and validity are questionable or the studied subjects are students which makes it difficult to generalize the results to professionals, since personality changes with major life events like the first job [Ahl18; SES11]. Therefore in Chapter 5 we propose hypotheses, how the different roles in a software development team should be staffed with respect to personality traits using the HEXACO personality theory (which is explained in detail in Chapter 2.1). In this study we interviewed 12 experts who work in the software engineering sector and analyzed the interviews following the Grounded Theory approach [GS10], thus the study is of qualitative character.

Chapter 4 investigates the correlation between personality traits and the maintainability of software. The literature review has shown that – until then – there were no studies on this subject. The study is of quantitative character.

The literature review also revealed that – beyond personality – gender also plays a role in the composition of teams. This is why we have conducted a study that examines the role of gender in Scrum teams, especially the role of the Scrum Master (see Chapter 6). Therefore we videotaped student scrum teams and applied principles of Grounded Theory for the analysis of the transcripts. This study is also of qualitative character.

Through the mix of methods, I have tried to take a comprehensive look at the issues under investigation.

Thus my work makes the following contributions:

- I present a Structured Literature Review which followed accepted standards and presents the research gaps on the subject "Relationship between Personality and Software Quality".
- My work provides practitioners with suggestions on how to staff different software development roles with respect to personality traits in order to build a good performing team.
- I provide a model which allows the prediction of maintainability by personality traits.
- I demonstrate the applicability and advantages of the Bayesian data analysis method for research in the area of investigating differences between groups also in the field of Software Engineering.

1.5. Previously Published Work and Shares of Work

- Parts of Chapter 3 were published in the Proceedings of the WorldCist'20 conference (`http://www.worldcist.org/`), pp. 766-777 under the title "The Influence of Personality On Software Quality - A Systematic Literature Review", publisher: Springer.
  Authors are Erica Weilemann and Philipp Brune.
  Erica Weilemann conducted the structured literature review. Guidelines were provided by Prof. Dr. Stefan Wagner. Two additional scientists were consulted to ensure the quality of data evaluation during the data extraction phase. Philipp Brune accompanied the work with helpful feedback.
- Concerning Chapter 4 data collection, analysis, interpretation and writing was performed by Erica Weilemann. Prof. Dr. Stefan Wagner suggested the usage of the Bayesian Data Analysis method.
- Chapter 5 was published as poster at the International Conference on Software Engineering 2019 (ICSE '19) in Montréal, Canada, Companion Proceedings pp. 252-253 under the title "A Winning Team - What Personality Has To Do With Software Engineering". CORE2018 ranking: A*.
  The author is Erica Weilemann, as the data collection, analysis and writing was performed by Erica Weilemann. The questionnaire was constructed together with Dr. Paula Bartel and Dr. Alexander Bartel for the project EVELIN `https://www.evelinprojekt.de/`. The two significant questions for this article and Chapter were inserted by Erica Weilemann. All the interviews used for data extraction in this Chapter were led by Erica Weilemann. The acquisition of the interviewees was also carried out by Erica Weilemann. The interviews were transcribed and analyzed by Erica Weilemann. To ensure the quality of the data evaluation two additional researchers (Prof. Dr. Stefan Wagner, Prof. Dr. Dany Meyer) were consulted. Prof. Dr. Stefan Wagner provided feedback for the writing. Prof. Dr. Philipp Brune and Tobias Ademmer

provided feedback for the design of the poster.

- Chapter 6 was published at the Australasian Software Engineering Conference 2015 (ASWEC ’15), Adelaide, Australia, pp. 3-7, under the title “Less Distress with a Scrum Mistress?: On the Impact of Females in Agile Software Development Teams” [WB15]. CORE2008 ranking: B, CORE2018 ranking Australasian.
  Authors are Erica Weilemann and Philipp Brune. Research design, data collection and data analysis as well as literature review was conducted by Erica Weilemann, writing was half conducted by Erica Weilemann, half by Philipp Brune. Philipp Brune invented the title.

If I have used the male form in my work, this is due to simplification and includes any form of the other genders as well.

One conclusion of my research work is that the term "software crisis" was formulated too hastily at the time. At that time one could only guess the speed of developments in the field of software engineering. Today you can feel it every day. The possibilities to develop software change so fast, for example through the introduction of new paradigms, new processes, new technologies – just think of quantum computers. This, of course, always results in new problems and questions and knowledge gained so far is questioned or no longer applicable, no longer up-to-date. This means that the process of finding new solutions for emerging problems in the field of software development is a continuous one and therefore one should not speak of a software crisis but rather of software development challenges. The process will always remain exciting and in my work I had the opportunity to look at a small part of it. I hope the reader will find these developments and studies as exciting as I did and still do.

Chapter 2

Background

Since we very often use the concepts "personality" and "maintainability" in our work, we will explain them in the following to create a common basis for our understanding of these concepts.

2.1. Personality

In Chapter 4 we analyze the correlations between personality and maintainability of software. We exposed our understanding of maintainability in the previous Chapter 2.2. This Chapter explains our choice of the HEXACO personality model and its emergence.

We chose the concept of personality because of its long-run tendency. The Amarican Psychological Association defines personality as:

Definition 2.1 (Personality)
“Personality refers to individual differences in characteristic patterns of thinking, feeling and behaving. The study of personality focuses on two broad areas: One is understanding individual differences in particular personality characteristics, such as sociability or irritability. The other is understanding how the various

parts of a person come together as a whole." [htt20]

These individual differences are often called personality traits.
The definition of "personality trait" is:

Definition 2.2 (Personality Trait)
"a personality trait refers to differences among individuals in a typical tendency to behave, think, or feel in some conceptually related ways, across a variety of relevant situations and across some fairly long period of time." [Ash18][p. 29].

Personality traits are stable for a "fairly long period of time", "it can probably be considered as a period of at least a few years" [Ash18][p. 31]. And "it is possible that even the rather stable, long-run tendencies of an individual might change considerably during the course of a lifetime" [Ash18][p. 31]. This shows that personality is a rather long lasting concept and tasks which suit better to people with certain prevalent personality traits do not necessarily have to be done by those people. Also other people are suitable for these tasks, they might just have to train longer because the tasks do not come naturally to them. Another aspect which has to be taken into account is that the measurement of personality has to be repeated from time to time. There are events in live which have a major influence on our personality and change it to a greater or lesser degree. One example is the entry into working life [SES11]. Deeper insights into the change of personality across lifetime can be found in [Ash18][p. 85ff] and [SES11].

In the following we present and justify our choice of personality theory.

2.1.1. Older Theory

As we have seen in Chapter 3 the research field of psychology in software engineering is associated with a strong increase in interest. Our study revealed that several personality type tests or inventories have been used for the studies in the previous 40-50 years.

The research field of psychology in software engineering is still an emerging field, although research in this field has been conducted already since the 1970s. Many of the personality type tests which have been used in the past for studies in the field of software engineering have been critizised due to their missing reliability and validity.

For example as we have seen in Chapter 3 one widely used personality type inventory in the field of software engineering is the Myers-Briggs Type Indicator (MBTI). This inventory is based on the theory of types of Carl Gustav Jung and assesses four different characteristics: Extraversion (E) vs. Introversion (I), Sensing (S) vs. Intuition (N), Thinking (T) vs. Feeling (F), and Judging (J) vs. Perceiving (P). People are assigned to one pole of each characteristic. Therefore 16 different personality types can be described with this model as there are 16 possible combinations of E/I, S/N, T/F and J/P [Ash18]. This personality type inventory is widely used in human resource management, and maybe this is the reason why it was widely used in the research field of software engineering. But this inventory is rejected by scientific psychologists because its reliability and validity is questionable [HLW03; Pit05].

Our aim was to find a personality test and theory which has proven to be valid and reliable and which brings interesting insights into the field of software engineering.

2.1.2. The Lexical Approach

Chapter 3 also showed another widely used personality theory in the field of software engineering – the Big Five Personality Factors or the Five-Factor Model.

The Big Five Personality Model resulted from a lexical approach. This means that the set of personality-descriptive adjectives which can be found in dictionaries is used as data basis for factor analysis. For this purpose a large sample of people is asked to "provide self-ratings on these adjectives" [Ash18, p.67]. Factor analysis is then conducted on the results of these self-ratings and returns major groups of personality traits [Ash18, p.67].

Table 2.1.: Examples of adjectives with high loadings on the Big Five dimensions obtained in lexical studies in the English language taken from [Ash18, p.69]

Dimension	**Adjectives**
Extraversion	talkative, extraverted, sociable, assertive vs. withdrawn, silent introverted, shy
Agreeableness	kind, warm, cooperative vs. cold, harsh, rough
Conscientiousness	organized, systematic, precise vs. careless, sloppy, unreliable
Emotional Stavility (vs. Neuroticism)	relaxed, unemotional, unexcitable vs. moody, jealous, anxious
Intellect/Imagination (Openness to Experience)	complex, unconventional, innovative, philosophical vs. simple, unintelligent, shallow, uninquisitive

The lexical approach is based on the lexical hypothesis. The lexical hypothesis assumes that all important characteristics that describe a person are represented by adjectives of the respective language colloquially.

Lexical studies in the English language led to the Big Five theory. The five dimensions of the Big Five theory are Extraversion, Agreeableness, Conscientiousness, Emotional Stability (vs. Neuroticism) and Intellect/Imagination (also known as Openness to Experience). Traits (adjectives) belonging to the same Big Five dimension are correlated, positively if they belong to the same pole and negatively if they belong to the opposite pole. Whereas traits belonging to different Big Five dimensions are generally roughly uncorrelated [Ash18, p.70]. Which also means that the five different dimensions are roughly uncorrelated. This fact is substantial for statistical analyses, e.g. when using Bayesian data analysis[1]. Adjectives describing these dimensions are taken from [Ash18, p.69] and can be found in Table 2.1.

Since the computing power at the time of the first lexical investigations

[1]the term which contains the interaction effects turns zero and can be neglected

of this kind (in the 70s) was not yet high enough to include all adjectives[1] in the factor analysis, the adjectives for these investigations were combined into clusters.

In the 1980s, the Big Five theory was a recognized theory among scientists to describe a person's personality using the 5 factors or dimensions. First questionnaires were constructed to measure these factors. Costa and McCrae constructed a questionnaire called the NEO Personality Inventory (NEO-PI), its revised form the NEO-PI-R and a shorter version, the NEO Five-Factor Inventory [CMP92],[Ash18, p.71]. In the 1980s and 1990s this questionnaire was a widely used tool in personality research. The five factors measured with the NEO-PI-R are broadly the same as described in the Big Five theory. There is one difference concerning the Intellect or Imagination factor. The Five-Factor model by Costa and McCrae labels it with Openness to Experience and takes away the intelligence factor as the researchers argue that intelligence is another construct and different from personality characteristics [Ash18, p.71].

Beginning in the 1980s, those lexical studies were conducted also in other languages. At that time computing power was much better compared to the beginnings of the lexical studies so that a list of adjectives with several hundred words could be used to conduct those studies, not "only" the 75 clusters. "By the early 2000s, lexical studies ... had been conducted in many languages", and revealed some differences from the English language results [Ash18, p.73]. Researchers found a similar set of six personality factors – across the different languages. This was the birth of the HEXACO model.

2.1.3. The HEXACO Personality Factors

The HEXACO model describes a person with the help of six personality factors. You find examples for adjectives describing these factors in Table 2.2.

[1]The original list of adjectives contained 4.500 adjectives. The list was reduced to approximately 2.800 adjectives by summary of synonyms. Then terms were excluded which were unfamiliar to study participants which resulted in a list of 1.700 adjectives. Similar adjectives were combined into 75 clusters which were then factor analyzed [Ash18, p.70].

Table 2.2.: Examples of adjectives with high loadings on the HEXACO factors taken from [Ash18, p.74]

Dimension	Adjectives
Honesty-Humility	sincere, honest, faithful/loyal, fair-minded vs. sly, deceitful, greedy, hypocritical, boastful
Emotionality	emotional, sentimental, fearful, anxious vs. brave, tough, independent, stable
Extraversion	outgoing, lively, sociable, talkative, active vs. shy, passive, quiet, reserved
Agreeableness	patient, tolerant, peaceful, mild, gentle vs. ill-tempered, quarrelsome, stubborn, choleric
Conscientiousness	organized, disciplined, diligent, precise vs. sloppy, reckless, lazy, irresponsible, absent-minded
Intellect/Imagination/Unconventionality (Openness to Experience)	intellectual, creative, unconventional, innovative, ironic vs. shallow, unimaginative, conventional

Five of the factors are very similar to those of the Big Five model. Nevertheless, there are differences here as well, since some adjectives are assigned to different traits in the two models. You can find examples for those differences in Table 2.3.

One main difference is the existence of a sixth factor – the Honesty-Humility factor. "This factor includes traits such as sincerity, fairness, and modesty versus slyness, deceit, greed, and pretentiousnes" [Ash18, p.75]. In English language studies these adjectives were mainly assigned to the Agreeableness factor but with small loadings [Ash18, p.75].

The Big Five model as well as the HEXACO model do not categorize people into personality types like the MBTI but speak of higher or lower loadings on different personality factors or personality traits. Thus no distinct personality type is derived but people are described in their complexity with the help of

Table 2.3.: Examples for different assignment of adjectives in the Big Five model and HEXACO model, excerpt taken from [Ash18, p.75]

Adjectives	**Big Five**	**HEXACO**
patient	high Emotional Stability	high Agreeableness
relaxed	high Emotional Stability	low Emotionality
anxious	low Emotional Stability	high Emotionality
tough	low Agreeableness	low Emotionality
irritable	low Emotional Stability	low Agreeableness
sentimental	high Agreeableness	high Emotionality

Table 2.4.: The six main personality factors of the HEXACO model with their facets taken from [Ash18, p.77]

Honesty-Humility	**Agreeableness**
Sincerity	Forgivingness
Fairness	Gentleness
Greed-avoidance	Flexibility
Modesty	Patience
Emotionality	**Conscientiousness**
Fearfulness	Organization
Anxiety	Diligence
Dependence	Perfectionism
Sentimentality	Prudence
Extraversion	**Openness to Experience**
Social self-esteem	Aesthetic appreciation
Social boldness	Inquisitiveness
Sociability	Creativity
Liveliness	Unconventionality

individual loadings on five or six main personality factors.

For measuring these six personality factors a questionnaire was developed – the HEXACO-PI and its revised form the HEXACO-PI-R [LA09a]. The HEXACO-PI-R assesses the six major personality factors as well as their narrower traits or facets. They are listed in Table 2.4.

The HEXACO-PI-R has proven to be valid and reliable [LA04]. And the HEXACO model is an accepted and frequently used theory among scientists in psychology. Therefore for our studies we will use the HEXACO PI-R as personality test. This test has been proven to be valid and reliable, like tests based on the Five Factor or Big Five theory. In contrast to the Five Factor theory, the HEXACO takes into account one additional personality facet - the Honesty-Humility factor. For our purposed investigations this could be a very interesting facet which has not yet been investigated in former studies in the field of software engineering. The German version of the HEXACO-PI-R which was used in our studies can be found in B.1 in Appendix B.

2.2. Maintainability

In order to clarify our understanding of maintainability we need a model which shows the characteristics by which we define maintainability and how they relate [FB15, p. 42f].

Software maintainability is a non-functional requirement and one facet of software quality. Several definitions of maintainability exist. "The ISO/IEC 25010 standard defines maintainability as follows: *Maintainability* is the degree of effectiveness and efficiency with which a product or a system can be modified by the intended maintainers. ISO/IEC 25010 2011" [FB15] "This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements." [iso17] Several aspects support the effectiveness and efficiency. Fenton and Bieman state that software which is "easy to understand, enhance or correct" is maintainable [FB15, p. 460].

"Maintainability is not restricted to code; it is an attribute of a number of different software products, including specification and design documents, and even test plan documents" [FB15, p. 461].

Very seldom, "documentation and models match the current version of the software system. Thus, almost always source code is the only reliable source of knowledge. Therefore source code is the central object of interest when we talk about program comprehension" [BSB08, p. 211].

Therefore, this study aims the internal measurement of maintainability and measures only the code. Further studies might investigate how the proposed aspects effect external measurement of maintainability, i.e. the measurement of the maintenance process.

But how can software maintainability be measured? Here also several theories and ideas exist. The reliability of these models have seldom been verified.

2.2.0.1. Decomposition Approach

Many researchers and practitioners built models which compose different subcharacteristics of maintainability to describe this software quality aspect - a so called "**decomposition approach**" [FB15, p. 443]. Boehm et al. e.g. decompose maintainability to the subcharacteristics testability, understandability, and modifiability. These "quality factors are still at too high a level to be meaningful or to be measurable directly. Hence, they are further decomposed into lower-level attributes called *quality criteria* or *quality subfactors*." [FB15, p. 443]. Boehm et al.'s model decomposes understandability into structuredness, self-descriptiveness, conciseness, consistency and legability [BBL76]. For these criteria they define direct metrics. The maintainability branch of the so-called "Boehm's Quality Characteristics Tree" [BBL76] can be found in Figure 2.1.

McCall's maintainability model is very similar. It decomposes maintainability into several subcharacteristics, which itself can be measured directly. McCall decomposes maintainability into consistency, simplicity, conciseness, self-descriptiveness, and modularity [FB15, p.444].

Newer approaches build similar models. Ludewig and Lichter [LL13] e.g. describe maintainability as a set of the subcharacteristics testability, changeability, and portability. These subcharacteristics itself are decomposed again. Ludewig and Lichter define maintainability as a facet of product quality. Other authors also include aspects of process quality to have an influence on maintainability (see e.g. [SSAO04], [CH09], [DWP+]).

The ISO/IEC 25010 decomposes maintainability into the subcharacter-

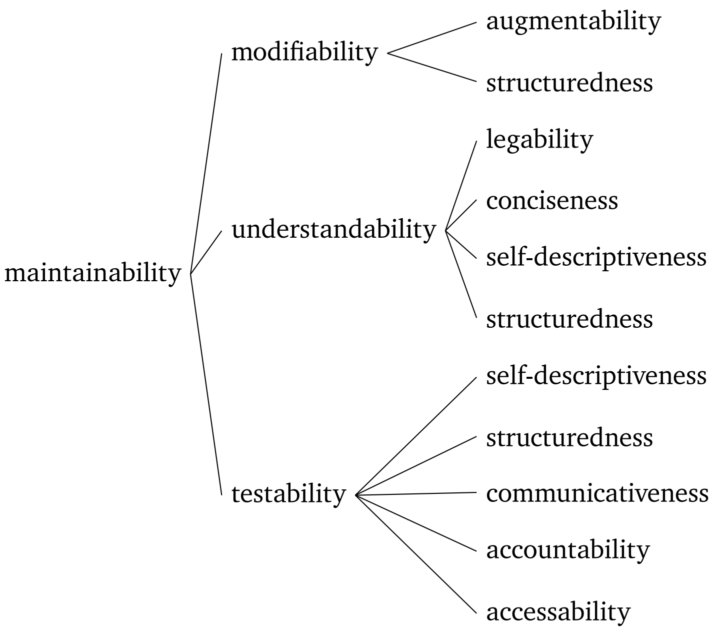


Figure 2.1.: Maintainability branch of the Boehm's Quality Characteristics Tree

istics modularity, reusability, analysability, modifiability, and testability [iso17]. Each of the subcharacteristics is defined itself [iso17].

2.2.0.2. Activity-Based Models

In recent time, another way to build quality models has emerged. Deissenboeck, Wagner et al. developed "activity-based quality models" [DWP+]. Their intention was to "make explicit the interrelation between system properties and the activities carried out on or with the system by the various stakeholders" [DWP+].

In their model, Deissenboeck et al. distinguish between activities carried out during software maintenance and characteristics of the software system or the company [DWP+]. They argue that such a model can be easily adopted by different companies with respect to their individual needs. "The set of relevant activities depends on the particular development and maintenance process of the organization that uses the quality model." [DWP+].

They also argue that the software system is not the only frame which differentiates the maintenance process and productivity of companies. "a plethora of other factors which include the skills of the engineers, the presence of appropriate software processes and the availability of proper tools like debuggers" also influence the productivity of a company [DWP+]. Thus in their model they do not speak of a software system but of a *situation*. The situation can be decomposed in what they call *facts* [DWP+].

An example for such a maintainability model is given in Figure 2.2.

The matrix visualizes the relationships between the *facts* and the *activities*. In order to express these relations, the entities have to be equipped with "fundamental *attributes* like *consistency*, *completeness*, *conciseness* or *redundancy*." [WDW]. Then *impacts* can describe the relation between activities and entities. "An impact is defined as a relation between an entity/attribute tuple and an activity where + expresses a positive and - a negative impact." [WDW]. Wagner et al. define the spelling as follows:

$$[\text{Entity e} \mid \text{ATTRIBUTE A}] \xrightarrow{+/-} [\text{Activity a}] \text{ [WDW]}$$

2.2.1. Software Metrics

Quality in general and software quality in particular are soft concepts. To make them tangible, one first step was to decompose the broad characteristics in order to find levels which can be measured. By measurement we achieve a possibility to "systematically and quantitatively" capture the software (system) [Dir13, p.247]. In this work we concentrate on static code analysis, which means we take a look only on written code, not on the process of software development, and the code is not executed. There are other techniques, e.g. software tests, which also can be used to measure certain aspects of software quality. Using testing techniques the code is executed. In this work we concentrate on a static code analysis.

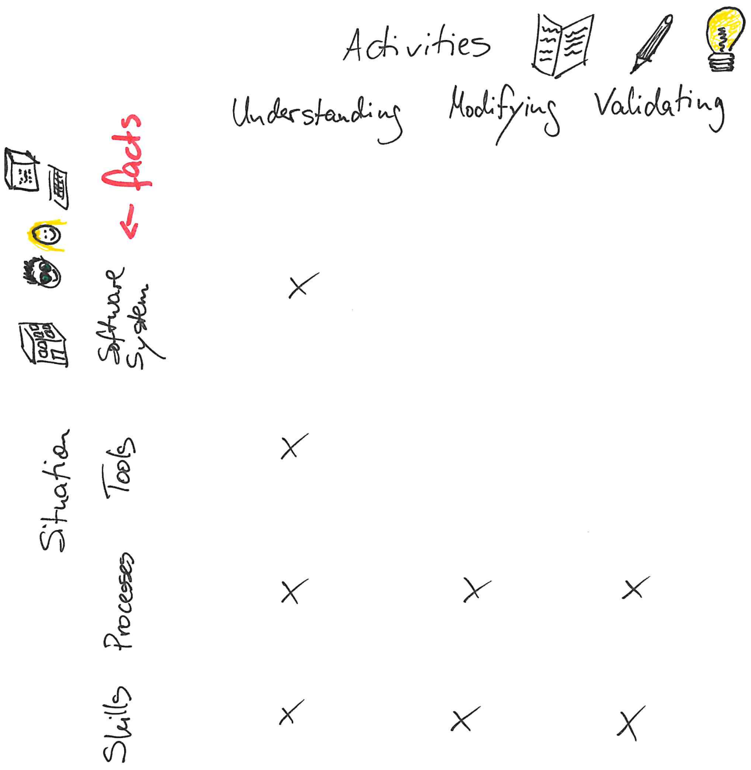


Figure 2.2.: Maintainability Matrix after Deissenboeck et al. [DWP+] where the x-axis represents activities and the y-axis the facts or properties.

We only glance at internal measures of software quality. ISO/IEC 25010 define it as "measure of the degree to which a set of static attributes of a software product satisfies stated and implied needs for the software product to be used under specified conditions

Note 1 to entry: Static attributes include those that relate to the software architecture, structure and its components.

Note 2 to entry: Static attributes can be verified by review, inspection, simulation and/or automated tools." [iso17].

2.2.1.1. Short History of Software Metrics

To measure software product quality the first metric used was LOC. The measurement dates back into the early 1960s. Over the time, LOC was used to measure different aspects of software "such as effort, functionality and complexity" [FN99]. With the increasing number of programming languages in the middle of the 1970s the need for individual metrics rose. The focus was on metrics for complexity, e.g. Halstead and McCabe, and programming language independent metrics for size, e.g. function points. "Work on extending, validating and refining complexity metrics (including applying them to new paradigms such as object oriented languages, e.g. Chidamber and Kemerer) has been a dominant feature of academic metrics research" since then [FN99].

Since the development of metrics, however, there has always been a critical discussion about their significance. The expressiveness of the Halstead metric has been questioned as well as the expressiveness of McCabe's Cyclomatic Complexity, Lines of Code and the Maintainability Index [OW].

The Chidamber-Kemerer-Metrics Suite was introduced in the beginning of the 1990s [CK91]. Since then "traces of the collinearity between some of the metrics and low ranges of other metrics" have been discussed [SPD+05]. The list of examples can be continued as long as you wish.

On the one hand, a critical examination of the metrics is necessary and useful. On the other hand, this makes it difficult to choose the "right" metrics in order to measure what is needed.

2.2.2. Our Set of Maintainability Metrics

To set up a model for measuring maintainability, we had to be aware of several basic conditions. As our model should be generally-valid and practicable, we oriented our procedure towards the “Goal-driven Measurement” [PGF96] and chose as basis Deissenboeck et al.'s activity-based quality model for maintainability [DWP+]. This model associates activities which are carried out during maintenance with system properties and thus directly links development and maintenance activities. This understanding is fundamental for good maintainability because "Maintenance begins with development. Thus maintainability is crucially shaped by initial development" [BSB08] (see also [Rei11]). The IEEE Computer Society demands – when describing basic developer human factors – the following: "it is essential to write software in a way that is easily understandable by humans or, at the very least, by other software developers." [IEE14, p.13-23].

Another requirement was the possibility of automatic measurement of the chosen metrics. The intention was to create an easy-to-use instruction which can be adopted with low effort. We were aware of the necessity of manual measurement in order to gain a comprehensive picture of the maintainability of the software. But we were as well aware of the fact, that in software developing companies there is less or even no personnel capacity for manual measurement of software quality aspects. "If measures should become daily routine, they have to be available at any time. To reach this goal, on the one hand they should be collected automatically and regularly and on the other hand they should be clearly presented" [BSB08, p.205]. So we decided to neglect the manual measurement.

2.2.2.1. Acitivities during Software Maintenance

Like April and Abran state, "Only in the 1970s did life-cycle models specific to the software maintenance process start to appear. In general, these first models were broken down into three steps: (1) understanding, (2) modifying, and (3) validating changes to software." [AA12]. Activities during maintenance are similar to activities during development. "Maintainers perform analysis, design, coding, testing, and documentation" [IEE14]. These can also be roughly clustered into the categories mentioned by [AA12]. But there are activities, processes and practices, which are unique to maintenance. Those are:

- Program understanding
- Transition
- Modification request acceptance/rejection
- Maintenance help desk
- Impact analysis
- Maintenance Service-Level Agreements and maintenance licenses and contracts

(Listing according to [IEE14].)

Understanding makes a huge part in software maintenance. "Program understanding consumes a significant proportion of maintenance effort and resources. At Hewlett Packard it was estimated that reading code (a fundamental element in comprehension) costs $200 million a year. Data from industry and other sources also indicates that about half of the total effort expended on effecting change is used up in understanding programs." [GT03]. Further the understanding of software can be influenced during its development. Thus in our evaluation we concentrate on 'understanding the software'. The IEEE Computer Society defines understanding as follows: "activities needed to obtain a general knowledge of what a software product does and how the parts work together." [IEE14].

2.2.2.2. The Situation - Representations of Software

Software can be represented in different ways: by its architecture, by its documentation, by its source code, by database tables, by build or installation scripts, by configuration or test scripts, by simulation and execution traces, by log and protocol files, by Wikis, FAQs, nutshells etc. [BSB08][p.]. All these artifacts help in understanding the software. In most cases, documentation is not up to date and thus source code is the only reliable source for understanding the system/software ([BSB08; GT03]). Thus in our investigation we concentrate on source code as our data basis.

2.2.2.3. Understandability of Source Code

One main demand - as mentioned above - is the understandability of source code. SWEBOK names the two main factors for understandability as structure and comments [IEE14, pp.13-23ff]. Other researchers complete the list with modularity [BR00; BSB08; GT03], the missing of code clones [BSB08; MRS+11], the consistency and meaningfulness of identifiers and names [BR00; GT03], a low complexity [BR00; GT03], a low level of nesting [GT03], design by contract [BSB08], the missing of unused code [BSB08], clarity [GT03], readability [GT03], simplicity [GT03], the existence of decomposition mechanisms [GT03], information hiding [GT03], the use of coding standards [GT03], indentation [GT03], spacing [GT03], boxing [GT03], shading [GT03], a set of tests [BSB08; Rei11], self-documenting code [Rei11], systematic reuse [Rei11], low coupling and high cohesion [Rei11].

As we see, concepts in the demands are mixed. Some authors speak of low complexity, others already refine this by demanding low coupling and high cohesion. We try to organize the terms in a structured way. We chose the partitioning of the activities following [AA12] and used as a basis for the situation the components listed by [BSB08]. The impacts are also described in the Table according to the suggestion of [WDW]

Table 2.5 shows an overview over the mentioned terms.

Table 2.5.: Visualization of our maintainability matrix structured according to the activity-based model from [DWP+]. On the left-hand side we find the facts according to [BSB08] and under the heading Maintainability we find the activities which are listed according to [AA12]. **U** = Understanding, **M** = Modifying, **V** = Validating

					Maintenance		
				possible measure	U	M	V
Situation	**Product**	architecture	design patterns | existence	separation of concerns	x	x	
			product line architecture | existence		x	x	
		documentation	glossary | detailed		x	x	
			glossary | complete		x	x	
			comments | existence		x	x	
			comments | detailed		x	x	
			design by contract | existence		x	x	
		source code	self-documenting code | existence		x	x	
			code clones | missing		x	x	
			not used code | missing		x	x	
			modularity | existence	coupling | low cohesion | high	x	x	

					Maintenance		
				possible measure	U	M	V
			systematic reuse | existence		x	x	
			complexity | low	level of nesting | low coupling | low cohesion | high	x	x	
			simplicity | high		x	x	
			clarity | high		x	x	
			readability | existence		x	x	
			readability | high	names | consistency/meaningful identifiers | consistency/meaningful coding standards | use of indentation | existence spacing | existence boxing | existence shading | existence	x	x	
			decomposition mechanism | existence		x	x	x

					Maintenance		
				possible measure	U	M	V
				information hiding | existence	x	x	x
				regression test baseline | existence			x
				set of tests | existence		x	x
			database tables	tables | existence	x		
			build (scripts)	scripts | existence	x		
			installation (scripts)	scripts | existence	x		
			configuration (scripts)	scripts | existence	x		
			test (scripts)	scripts | existence	x		
			simulation (traces)	traces | existence	x		
			execution (traces)	traces | existence	x		
			log files	files | existence	x		
			protocol files	files | existence	x		

					Maintenance	
			possible measure	U	M	V
		Wikis	Wikis | existence	x		
		FAQs	FAQs | existence	x		
		nutshells	nutshells | existence	x		
	Infrastructure	tools	compiler | existence		x	x
			debugger | existence		x	x
			code navigation tools | existence	x	x	
			pretty-printer | existence	x	x	
			code-rule-checker | existence		x	
			profiler | existence		x	x
			code-coverage-tools | existence		x	x
			metric tools | existence		x	x
			graph-visualization-tools | existence	x		x
			graph-verification-tools | existence	x		x

				Maintenance		
			possible measure	U	M	V
			architecture-analysis-tools | existence	x		x
			automatic documentation support tools | existence		x	

In the following we are going to address the shaded fields because they concentrate on source code and documentation which (in parts) can also be measured with the help of the source code only.

2.2.2.4. Suitable Metrics

In order to measure the listed requirements for understandable software, we found the following metrics to be helpful in obtaining a picture if the developed software is understandable. We list them together with sources which encourage the use of the respective metric or concept.

- Comments | Existence
  - Number of Comments
  - Comments Ratio
- Number of Code Clones [FB99], [Wag13], [Vog12], [HKV07]
- Dead Code [Wag13], [Vog12]
  - Code Coverage
- Complexity
  - Size
    - LOC (lines of code; as norming factor) [EWIM11]
    - NOM (number of methods in a class, = MPC) [LM06], [LH93], [FB99]
    - LOC/Method [LM06], [FB99]
    - LOC/Class
    - Number of Parameters/Class [FB99]
    - Faults/KLOC [FB15]
  - Structural Complexity
    - RFC (Response for Class = number of local methods + number of methods called by local methods) [LH93]

        - Nesting Depth [FB15]
        - PF (Polymorphism Factor) [BM96]
- Architecture and Structure
    - Inheritance
        - DIT (Depth of Inheritance Tree) [LH93]
        - ANDC (Average Number of Derived Classes) [LM06]
        - AHH (Average Hierarchy Height) [LM06]
        - NOC (Number of Children) [LH93]
        - MIF (Method Inheritance Factor) [BM96]
        - AIF (Attribute Inheritance Factor) [BM96]
    - Modularity
        - Cohesion and Coupling
            - CBO [DR01], [HKV07]
            - CALLS/Method (Coupling Dispersion) [LM06]
            - FANIN [HKV07]
            - FANOUT/operation call (Coupling Dispersion) [LM06], [HKV07]
            - DAC (Data Abstraction Coupling) [LH93]
            - LCOM4 [LH93]
- Reuse
    - Reuse Density
- Tests
    - Code Coverage

2.2.3. Tools

For our purpose, a tool for measuring software metrics has to meet several conditions. Our research is restricted to Java code as Java was and is the prevalent programming language [Put17; www18]. In order to simplify and streamline the measurement process in the companies, we use only one tool and not a combination of tools. Thus the tool should measure as many metrics as possible that we need. We concentrate on metrics on class level because classes represent the basic idea of object orientation. Another restriction was the license of the tool. It should be freely available.

In our search we discovered the research work of Lincke et al. [LLL]. They conducted a similar search of static code analysis tools. From their work we found that the tool VizzMaintenance[1] computed most of the metrics we needed compared to the other tools. Our own search resulted in 19 tools (see Table 2.6). Three of these were not available for the entire duration of the investigation. Of two tools, only very outdated versions were available. Some of the remaining 14 tools concentrated only on code clones and were not suitable for other aspects which we wanted to measure. Two of the tools were almost identical. Thus the final list of our tools was very similar to the list of Lincke et al. [LLL]. Therefore we decided to do our measurement with the Eclipse plugin VizzMaintenance [LLL]. The VizzMaintenance User Guide which can be found in the Eclipse Help section describes very detailed which metrics are measured and why. Lincke and Löwe also explain very detailed the implementation of the metrics in [LL07].

We analyzed which metrics can be measured by the respective tools. Table 2.7 shows our results.

2.2.4. Further Research

In our research, metrics of the source code are evaluated. Some researchers, e.g. Genero et al. [GPM04] propose to gather measurement data even in earlier stages, i.e. in the design phase by analysing UML diagrams. Further

[1] `http://www.arisa.se/vizz_analyzer.php`

Table 2.6.: List of measuring tools

- JHawk [1]
- CodePro AnalytiX [2]
- PMD/CPD [3]
- Clone Digger [4]
- Clone Doctor [5]
- ConQAT [6]
- IntelliJ IDEA Community Edition [7]
- UCDetector [8]
- VizzAnalyzer Metrics Suite [9]
- NDepend [10]
- SonarQube [11]
- Jarchitect [12]
- Analyst4J
- Eclipse Metrics plugin [13]
- CCCC [14]
- Dependency Finder [15]
- Eclipse Metrics [16] [17] [18]
- Understand for Java [19]
- Google Testability Explorer [20]

1 http://www.virtualmachinery.com/jhawkmetricslist.htm

2 https://developers.google.com/java-dev-tools/codepro/doc/

3 http://pmd.sourceforge.net/pmd-4.3.0/cpd.html

4 http://clonedigger.sourceforge.net/

5 http://www.semdesigns.com/products/clone/

6 https://www.cqse.eu/en/products/conqat/overview/

7 http://www.jetbrains.org/pages/viewpage.action?pageId=983211

8 http://www.ucdetector.org/

9 http://www.arisa.se/compendium/node110.html

10 http://www.ndepend.com/docs/code-metrics

11 http://docs.sonarqube.org/display/SONAR/Metric+definitions

12 http://www.jarchitect.com/Metrics#MetricsOnApplication

13 http://sourceforge.net/projects/metrics/

14 http://cccc.sourceforge.net/

15 http://depfind.sourceforge.net/

16 http://eclipse-metrics.sourceforge.net/

17 http://metrics.sourceforge.net/

18 http://realsearchgroup.org/SEMaterials/tutorials/metrics/

19 https://scitools.com/feature/metrics/

20 https://code.google.com/p/testability-explorer/

research could be done in investigating the differences in the outcome of these results. Are there differences? To which degree? Are there monetary losses?

Table 2.7.: Measuring tools and their implemented metrics

	Number of Comments	Comments Ratio	Code Clones	Dead Code	Code Coverage	Lines of Code (LOC)	Number of Methods	LOC/Operation	Number of Classes	LOC/Class	Number of Parameters	Number of Parameters/Class	Faults/KLOC	Polymorphism Factor	Response for Class (RFC)	Nesting Depth	Depth of Inheritance Tree (DIT)	Average Number of Derived Classes (ANDC)	Average Hierarchy Height (AHH)	Number of Children (NOC)	Method Inheritance Factor (MIF)	Attribute Inheritance Factor (AIF)	Coupling Between Objects (CBO)	Calls/Operation	FANIN	FANOUT	Data Abstraction Coupling (DAC)	LCOM (4)	Reuse Density	Afferent Coupling	Efferent Coupling	Nested Block Depth	LCOM	Average Block Depth	Average Depth of Inheritance Hierarchy
JHawk	no longer available in 2018																																		
CodePro AnalytiX	no longer available in 2018																																		
PMD/CPD			x																																
Clone Digger			x																																
Clone Doctor	Disadvantage: different versions depending on Java 1.5, 1.6 etc.																																		
ConQAT	no longer available in 2018																																		
IntelliJ IDEA Community Edition	Code Clones only in paid version																																		
UCDetector	finds dead code but only in public Java code																																		
VizzAnalyzer Metrics Suite						x	x								x		x			x			x				x				x				
NDepend	x	x				x	x		x		x	x			(x)		x			x										x	x		x		
SonarQube	x	x	x			x			x	bad for complexity measurement because McCabe is used																									
Jarchitect	very similar to NDepend																																		
Analyst4J	only outdated version																																		
Eclipse Metrics plugin	last Update: 2013																																		
CCCC	see [LLL]																																		
Dependency Finder							x				x	x					x								x	x				x	x				
Eclipse Metrics						x	x		x		x						x			x										x	x	x	x		
Understand for Java	x	x					x	x			x						x	x															x		
Google Testability Explorer																																			

CHAPTER 3

State of the Art

The Influence of Personality on Software Quality – A Systematic Literature Review

Software quality is an important factor for reducing costs or increasing sales of a (software developing) company [Cha05]. Humans are involved in the whole software development process. This suggests that human factors have an influence on the result of the development process. We aimed to understand which distinct human factors - especially which personality traits - have an influence on which distinct software quality attributes. For this purpose, we conducted a Systematic Literature Review following the guidelines of Kitchenham and Charters [KC07].

3.1. Background

For quite some time now, the claim has raised for taking into account human factors in information systems (IS) research. Early studies demanded a shift of the perspective away from “human has to conform to computers” to “computer has to serve the human needs” [Hop86]. In the early 80s, Shneiderman presented four approaches to “include human factors during system design” [Shn81]. Him too, he emphasized the need to take into account human factors during the system or software development process. Hanenberg mentions “an urgent need to consider human factors when reasoning about programming language and software engineering artifacts” [Han10]. Lately rose a request for communities which discuss and investigate the influence of human and social factors on software development problems and software engineering approaches [JMT05].

The interest in human factors in the field of software engineering is not new. There is a huge amount of literature in this field and researchers have already investigated, which software engineering topics have been addressed and which human factors were taken into account. Cruz et al. e.g. investigated “the methods used, topics addressed, personality tests applied, and the main findings produced in the research about personality in software engineering” [CdM+11]. The aim of Pirzadeh was to “identify and characterize human factors influencing the software development process from development lifecycle and software management perspectives” with the help of the existing literature [Pir10]. And Maier investigated “personality traits on distinct hierarchical levels in three fields of information systems research” within a very restricted source of literature [Mai12]. We found even more literature reviews on this topic. But they did not satisfy the DARE criteria given in [HG08] and because of this will not be listed.

All these literature reviews covered a broad area: software engineering in general. Our specific interest was the impact of human factors or personality on the outcome of a software engineering process – the produced software. The listed literature reviews did not answer this question.

Precisely software quality is a very substantial factor for the success of a software producing company. Customer satisfaction can lead to subsequent orders which leads to increasing incomes for the company. On the other hand, a company could reduce costs. If we only consider software maintainability, “maintenance makes 40%–80% of the software costs” [Gla01]. An improvement of software maintainability subsequently leads to a reduction of maintenance costs. If human factors affect software quality aspects like maintainability, performance, and others, software producing companies could save money by assigning the right tasks to the right people. In addition, the satisfaction and motivation of the employees could be increased because they are more pleased with the results they produce.

3.2. Review Questions

Our specific interest is the influence of the personality of a software engineer on software quality attributes.

The general research question is:
RQ: “How can the impact of the personality of a software engineer on software quality be described?”
We devided this question into two supporting sub-research questions:
RQ1: Which personality types do software engineers have?
RQ2: Which attributes of software quality are affected by personality?

These questions were structured according to the recommendations of [PR08] using the **P**opulation **I**ntervention **C**omparison **O**utcomes **C**ontext (PICOC) criteria.

The population are software engineers, not restricted to companies or universities.

The intervention is the personality or personality type of the software engineer.

For RQ2, the personality type is also the comparison.

Outcomes of RQ1 are software engineering characteristics and personality types.

Outcomes of RQ2 are software quality attributes.

The context of both research questions is not limited.

3.3. Review Methods

3.3.1. Data Sources and Search Strategies

3.3.1.1. Data Sources

As data sources, we chose three digital libraries as well as results of "manual" search. The three digital libraries were taken from [Tur]. They are IEEE Explore[1], ACM Portal[2], and ScienceDirect[3]. Search was conducted in this sequence. So we looked for duplicates in this sequence, too. The question type is no question with a primary technical focus, so Kitchenham's question types [KC07] do not match. The Australian NHMR Guidelines [NHM00] are more applicable. Type five matches: "Identifying whether a condition can be predicted". With the results, a prediction of software quality attributes should be possible by knowing the personality type of the corresponding software engineer. The question is meaningful and important to practitioners as well as researchers, because it will be possible to predict software quality attributes through personality type testing and it enables an optimization of forming a software development team with the purpose to increase software quality. For each question, according to Kitchenham et al. [KC07] we identified population, intervention, comparison, outcomes, and context. We took the resulting phrases as basis for searching synonyms. In our search terms, we used the operators OR and AND as well as the wild card *.

Another source were bibliographies of articles which were results in the first phase or already read in another context. For one article, we had to contact the author because it was not accessible online.

[1]http://ieeexplore.ieee.org
[2]http://dl.acm.org
[3]http://www.sciencedirect.com

3.3.1.2. Search Strings

We conducted the search upon RQ1, which is broader. The resulting papers include papers that concern RQ2. During the data collection phase, we extracted the data that were distinct for RQ2. Because of individual characteristics of the digital libraries, the search terms were customized to each digital library. The resulting search terms are the following:

- For IEEE Explore and ACM Portal: *((("information system" OR comput* OR IT OR IS OR program* OR system) AND (engineer OR professional* OR personnel OR people OR practitioner* OR producer OR role)) AND (personalit* OR "cognitive style" OR personality trait OR "human factor"))*. This search resulted in 14.797 articles in IEEE Explore and 5.045 articles in ACM Digital Library.
- For ScienceDirect: we conducted two searches in ScienceDirect. The first search string was: *("information system" OR comput* OR "IT" OR "IS" OR program* OR system) AND (engineer OR professional* OR personnel OR people OR practitioner* OR producer OR role) AND (personalit* OR "cognitive style" OR personality trait OR "human factor") AND NOT (education OR "HCI" OR "human computer interaction" OR agent OR avatar OR robot* OR game*) AND LIMIT-TO(topics, "social psychology,individual difference,personality trait,personality,emotional intelligence") [All Sources(Computer Science,Psychology,Social Sciences)]*. This search resulted in 819 articles. From these, no article was directly linked to the topic. So we conducted a second search with another search string. We held this string very simple. The second search string was: *"software quality" AND "personality"[All Sources(Computer Science,Psychology,Social Sciences)]*. Here, the search delivered 758 articles.

We did not limit the results by the year of publication.

3.3.2. Study Selection

3.3.2.1. Including and Excluding Criteria

As we were looking for the impact of the personality of a software engineer on software quality, all articles which describe the personality of a software engineer and software product quality were included. The search strings indicate that we obtained a huge amount of articles and thus had to apply excluding factors.

In one searching engine, it was possible to exclude some topics already in the search. But where this was not possible, we had to narrow down the amount of not directly linked studies. For this purpose we defined excluding criteria. Those were *human-computer-interaction; team performance; technical issues (reuse, automation, architecture; nothing to do with software engineering; school concepts; aviation, space flight systems, nuclear power plants; engineering tools; biometry; educational concepts; learning systems; knowledge management; health care; other engineering disciplines (e.g. mechanics); gameplay; gamification; biological/pharmaceutical issues; machine learning; learning styles; intelligent systems; virtual humans; virtual agents; patterns; CAD; automated knowledge acquisition; virtual reality; neural networks; music; cognitive engineering; automotive; car IT; indices/table of contents of conference proceedings; decision support systems; creativity; trust in virtual teams; tools for distributed software teams; robots; language is not English or German*.

3.3.2.2. Study Selection Process

The study selection process comprised three phases. In the first phase we scanned the titles for relevance. If titles showed no relevance for the research questions, the study was excluded. If it was not clear, the study was not excluded. In the second phase we scanned the abstracts of the remaining papers and checked if the paper was relevant for our research questions. If not, the study was excluded. In the third phase – the final selection phase – we read the complete paper and decided whether its content is relevant or not.

3.3.3. Quality Assurance of the Studies

For each study 22 quality criteria were checked. Those were questions which could be answered with yes or no. Only in one case, a choice between yes, moderately, and no was possible. These criteria were e.g. “Does the study report clear, unambiguous findings based on evidence and argument?”, “‘Were the basic data adequately described?”, “Do the researchers explain the consequences of any problems with the validity/reliability of their measures?” and “Is the paper well/appropriately referenced?”. We defined the criteria following [BBHS]. Our quality measure for each paper was the rate of positively answered questions. 0% means very poor quality, 100% means very high quality/complies with all our quality requirements. Papers with a quality level less than 50% were excluded from the study. The sheet we have used for the evaluation can be found in Appendix A – Figure A.1.

3.3.4. Data Extraction and Synthesis

For each study, several information were selected in an Excel datasheet. Those concerned some general information on the paper like title, author, paper id[1], year, and country of the study. But they also contained information on the type of study, data collection methods, number of study subjects, type of subjects in the study (students or practitioners/experts), role of the subjects in the software development process, type of personality type test, the software development process model, whether the development process was agile, the team size, the programming language, and whether the findings are relevant for the research questions or not (is the work directly linked to the research questions or partly or not at all). Also we captured if group attributes were taken into account and which ones, if and which cognitive model or style was taken into account, which software quality

[1]first two letters of the first author’s surname + year of publishing + first four letters of the title, e.g.: M. Petticrew and H. Roberts. Systematic reviews in the social sciences: A practical guide. John Wiley & Sons, 2008 –> paper id: Pe2008Syst

attributes were taken into account, and if motivation was taken into account. We collected the key findings for each paper in the table as well as the research questions.

3.3.5. Threats to Validity

Investigator Bias: This systematic literature review was conducted by an individual researcher. Therefore, the probability of validity threats is higher than if several researchers would have conducted the systematic literature review together. Because of this, we implemented several (preventive) measures. During all phases, the individual researcher consulted a second researcher who has gained experiences with systematic literature reviews. Interim results and next steps were discussed. In addition, in the data extraction phase, the individual researcher called in a separate group of researchers. The purpose and results are explained in *Data Extraction*.

Publication Bias: Since this systematic literature review did not intend to find any positive effects of treatments, this bias is low in this survey. The intention of this systematic literature review was of exploratory nature – investigate the link between personality type and software quality. In addition, the search was not restricted to journal articles only. The individual researcher took into account several different sources of articles, e.g. proceedings, books, and gray literature.

Threats to the Identification of Primary Studies: To assure a high coverage of primary study inclusion, the researcher chose to use ACM Digital Library, IEEEXplore, and ScienceDirect as the main sources for automated search. These digital libraries are well known and among the most common databases for research in this area. In addition, the researcher conducted a parallel manual search. Sources were e.g. the bibliography of articles, articles that were read in another context, and discussions with other researchers. Furthermore, the author customized the search strings for the different digital libraries to ensure as many relevant results as possible.

Threats to Data Extraction: As data extraction is one of the most important phases for the presentation of the results, the researcher focused special attention on the validation of the method. For this purpose, the researcher asked other researchers from the same faculty to extract data from randomly chosen articles. 5% of all articles remaining after phase three were chosen randomly and sent to the researchers. They also received an Excel sheet of the same form the individual researcher used to extract data. They also received the Excel forms to asses the quality of the papers. The individual researcher did not send a special code book to the other researchers which accurately explains all the variables used in the sheet. The individual researcher explained in an e-mail the most important variables and how to handle the Excel sheet. With the help of the returned answers the individual researcher calculated the percent agreement and Krippendorff's Alpha [Kri07] for the variables paper id, title, year, country, style of study, subjects of the study, number of the subjects, type of personality type test, software quality attribute, weight of the findings for the research question, and key findings. The percent agreement of all variables was 79.8%. This is a very high agreement. Krippendorff's Alpha for the distinct variables is given in Table 3.1.

Table 3.1.: Krippendorff's Alpha for the distinct variables of the data extraction form.

Paper Id	Title	Year	Country	Style of Study	Study Subjects	Number of Subjects	Type of Personality Type Test	Software Quality Attribute	Weight of Findings for Research Question	Key Findings
1.0000	1.0000	1.0000	0.8777	0.7792	0.5487	0.8571	0.7536	0.4070	0.5143	0.8828

These results show moderate to substantial agreement [LK77].

3.4. Results

At the end of our study, we did a lot of paperwork. The dimensions of this work are shown in Table 3.2 – this table shows the number of search results after each phase of searching and selecting.

From Figure 3.1 we see that starting from 1978 the MBTI was very prevalent in studies related to Software Engineering or Computer Science. Other tests then till 1989 use predominantly Murrey's Variables of Personality. The use of Keirsey(-Bates)-Temperament-Sorter can be observed sporadically since 2005. This theory emerged in the end of the 70s and is based on the MBTI.

MBTI is used constantly since 1979. For Keirsey and MBTI there is a huge industry behind Keirsey (`https://www.keirsey.com/`) and the MBTI (`https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1` and other companies). These two theories are widely used in business settings. "However, one shortcoming of the MBTI is that it loses a great deal of precision by describing people in terms of only two levels of each characteristic rather than in terms of a more specific score on each characteristic." [Ash18][p. 47]. Also it's validity and reliability were questioned [HLW03].

The use of a personality type theory related to the Big Five (Big Five, NEO-PI, IPIP, Five Factor, etc.) started in 1998 and since then its use increased [1]. Other personality type tests which were used in this period of time comprise Rosenberg's Self-Esteem Scale, Judge's Generalized Self-Efficacy Scale, Levenson's Locus of Control Scale, Eysenck and Eysenck's Neuroticism scale; Schwarzer and Scholz's instrument; Fifteen Factor Questionnaire Plus and 16 Personality Factors.

[1] some of the authors did not distinguish between the Big Five theory and the Five Factor Model. Thus we built only one category for all related personality type theories which comprise five dimensions.

Table 3.2.: Number of search results after the three phases, including manual search

Digital Library	Number of Search Results	Number of Duplicates	Number of Relevant Articles After Phase 1 + 2	Number of Relevant Articles After Phase 3
IEEE Explore	14.797	0	278	52
ACM Digital Library	5.045	109	176	40
ScienceDirect	758	0	50	11
Manual Search	27	0	27	6
Total:	**20.627**	**109**	**531**	**109**

Till April 2016, we did not find a study which used the HEXACO Personality Inventory, also not the revised form.

Figure 3.2 reveals that the number of studies including personality factors increases since the beginning of the 2000s. This concerns especially the number of quantitative studies. There was a slight increase of qualitative studies round 2010.

Figure 3.3 shows that the number of studies reporting experiments with students is increasing since the beginning of 2000. The Figure also shows that there is a comparably high number of studies carried out with experts. The term "others" includes e.g. literature, as for literature reviews the studied objects are articles, books etc.

The most studied role was programming (25 %), as is shown in Figure 3.4. The second most studied role was testing (6 %). Many studies do not distinguish between roles, which is indicated by "mixed".

Some studies took into account "only" performance where performance - if at all - is completely individually defined. Only a few studies incorporated software quality attributes. The involved quality attributes were:

– bug density (defects per 1000 lines of code)

- code design (code efficiency and code readability)
- code productivity (code lines and accuracy)
- (the two aforementioned criteria were subjective evaluations)
- cognitive weights (software complexity)
- complexity
- decomposition and modularization (number of modules and coupling)
- design and correctness (grade)
- design time
- efficiency
- encapsulation
- functionality (number of satisfied requirements)
- generalization
- information hiding
- ISO/IEC 9126 suitability of the product (sub-characteristic of functionality)
- design adequacy (ratio of number of design modules evaluated to function adequately to the number of design)
- implementation adequacy (ratio of the number of source code units evaluated to function adequately to the number of source code units)
- functional implementation completeness (one minus the ratio of the number of functions detected as missing during evaluation to the number of functions described in the requirements)
- functional implementation coverage (one minus the ratio of the number of functions detected as missing or incorrectly implemented during evaluation to the number of functions described in the requirements)
- functional specification volatility (ratio of number of requirements changed after baseline to the number of requirements)

- gate cost
- issue lead time (bug fixing time)
- maintainability
- mean number of design operations between design errors
- memory consumption
- number of defects
- programming style (guidelines defined at the subject website)
- readability
- reusability and (number of reused modules)
- run time
- specialization
- specification conformance
- structural complexity (LCOM, DIT, CBO, NOC, RFC, WMC, total lines of code, total lines of comments, total blank lines)
- testability (number of defects detected by the automatically executed test case set)
- understandability
- usability

Only two studies investigated factors of maintainability [AGE95; BM96]. Brito e Abreu et al. investigated "the impact of Object-Oriented design on software quality characteristics" [AGE95]. They dealt with "the quantitative evaluation of design attributes of object-oriented software systems" because they "believe that these attributes can express the quality of internal structure" and thus influence the maintainability of software [BM96]. But no people involved in the software development process were included as influencing factor on maintainability which shows the importance of my study.

For RQ1, the topics of the related papers can be clustered into two main topics:

1. Researchers examined which personality types match distinct software engineering roles.
2. Researchers examined the distribution of personality types among staff in the software engineering industry and in comparison to the population of the country.

Personality types for distinct software engineering roles (RQ1.1.) were the following ones:

Project Manager

Agreeableness: emphasize ease of use and vendor support [BH]
Conscientiousness [BT]
Conscientiousness: consider cost as crucial evaluation factor [BH]
Extraversion: attach importanced to vendor support and ease of implementation [BH]
Neuroticism: put emphasis on functional aspects and reliability [BH]
Openness to Experience [BT]
Openness to Experience: focus more on the ease of customization [BH]

Team Leader

Ability to carry ideas through in association with the team [AJM06; YEMT05]
Ability to define appropriate plans for achieving goals [YEMT05]
Ability to provide ideas (large and small scale) [AJM06; YEMT05]
Analytical [AJM06; YEMT05]
Customer service [AJM06]
Decision making [AJM06]
Empathy [AJM06]
Environmental knowledge [AJM06]
Environmental orientation [AJM06]
Negotiation skills [AJM06]
Risk management [AJM06]
Sociability [AJM06]
Stress tolerance [AJM06]

Support the team [AJM06; YEMT05]
Take responsibility for a team [YEMT05]

Technical Leader

Analytical [YEMT05]
Capable of taking a project forward [YEMT05]
Prepared to take advice to seek out the best route forward [YEMT05]
Teamoriented [YEMT05]

System Analyst

Agreeableness [RMSA12]
Extraversion [CA10b; RMSA12]
Extraversion and Intuition [CT98]
Feeling [CA10b]

Architect/System-/Software Designer

Ability to coach [Ber08]
Ability to listen [Ber08]
Ability to motivate [Ber08]
Agreeableness [RMSA12]
Analytical [AJM06; YEMT05]
Attention to detail [Ber08]
Capable of taking a project forward [YEMT05]
Decision making [AJM06; Ber08]
Discipline [AJM06]
Empathy [AJM06]
Environmental orientation [AJM06]
Independence [AJM06]
Introvert and Sensing [CT98]
Intuition [CA10b]
Intuition and Thinking [CT98]
Open mind [Ber08]
Openness to Experience [RMSA12]
Self organization [AJM06]
Sensing and Judging [CT98]

Stress tolerance [AJM06]
Take advice to seek out the best route forward [YEMT05]
Teamoriented [AJM06; YEMT05]
Tenacity [AJM06]
Thinking [CA10b]
Thinking and Perceiving [CT98]
Up and down communication [Ber08]
Visionary [Ber08]

Programmer

Agreeableness [RMSA12]
Analytical [AJM06]
Decision making [AJM06]
Discipline [AJM06]
Empathy [AJM06]
Environmental orientation [AJM06]
Extraversion [RH07; RMSA12]
Independent [AJM06]
Introversion [CA10b]
Intuition and Thinking [CT98]
Openness to Experience [RH07; RMSA12]
Self organization [AJM06]
Sensing [CA10b; CT98]
Stress tolerance [AJM06]
Teamwork and cooperation [AJM06]
Tenacity [AJM06]
Thinking [Bis95; CA10b]

Tester

Conscientiousness [RMSA12]
Discipline [AJM06]
Empathy [AJM06]
Environmental orientation [AJM06]
Extraversion for exploratory testing [SNA09]
Independence [AJM06]

Intuition+Thinking for code review tasks [DG07]
Judging [AJM06; CA10b]
Locus of Control [Bis95]
Openness to Experience [RMSA12]
Self organization [AJM06]
Sensing [CA10b]
Stress tolerance [AJM06]
Teamwork and cooperation [AJM06]

Maintainer

Customer service [AJM06]
Discipline [AJM06]
Empathy [AJM06]
Environmental orientation [AJM06]
Perceiving [CA10b]
Self organization [AJM06]
Sensing [CA10b]
Stress tolerance [AJM06]
Teamwork and cooperation [AJM06]
Tenacity [AJM06]

Table A.1 shows a detailed list of the results for RQ1.1. and can be found in Appendix A.

All papers, which examined RQ1.2. report that the distribution of personality types of software engineering staff differs from the distribution of personality types in the population of the country. It seems that people with special personality traits feel attracted to this job. One paper also reports a change of distribution over a long period of time in the software engineering industry [VCPR12]. One reason is that the method of software development has changed over the past thirty years. Table A.2 shows a detailed list of the results for RQ1.2. and can also be found in Appendix A.

The search results that are related to RQ2 can be split into two groups:

1. Personality traits that are important for a high team performance and
2. personality traits that contribute to improve software quality.

In many papers where the term software quality was used there was no precise definition of what was exactly meant by software quality, i.e. which quality attributes were considered. Or it was not told, which measures were taken into account to determine the respective quality attribute. In many cases, the only metric that was given was "grade point average", but the determinants which lead to the grade were not given. In some cases, software quality was substituted by personal performance, which often was measured by grade point average or external assessment of a superior.

Table A.3 shows the search results which are linked to RQ2.1. and can be found in Appendix A.

Results of RQ2.1. mainly cover three areas. We found *distinct personality traits* which have a direct influence on team performance. But we also found personality traits which influence factors like communication, which in turn have a direct influence on team performance. Another area is the *optimal distribution of different personality types* in a software development team. The third large issue is the influence of *expertise* on team performance. These links are illustrated in Figure 3.5.

Table A.4 shows the search results which are linked to RQ2.2. and can be found in Appendix A.

The search results for RQ2.2. can be split into several topics which have an influence on software quality or personal performance (see Table 3.3).

Some papers show the direct impact of distinct **personality traits** on performance or software quality attributes. Other papers argue for **experience** as a main influencing factor of performance. Some papers only speak of **personality in general** as an influencing factor on performance. One paper mentions the influence of individual differences in design considerations on memory consumption and runtime [Pre+99]. Design considerations are creative work and thus influenced by personality [KWB96]. Another group of papers links **cognitive abilities** and software quality. **Self-efficacy** and **values** have been found to be reliable predictors for performance and competence. There was one group of papers that we could not cluster. These papers state in general the influence of human factors or skills on software quality and performance.

We also found papers which rejected a link between personality traits and software quality or performance. In his investigations, Gallivan could not find a significant link between "Openness to Experience" and job performance [Gal04]. Darcy and Ma argue that "little research done to date has identified the personality profile for an excellent programmer" [DM05].

Table 3.3 shows a summary of the above described findings to RQ2.2.

Many quantitative studies were conducted. For RQ1, the majority of the studies

Table 3.3.: Summary of influencing factors on software quality or performance. (t.b.c.)

Distinct personality traits and their impact on performance or software quality attributes			
This	*has impact on*	*reports who*	*group*
Extraversion	Exploratory testing	[SNA09]	Distinct personality traits and their impact on performance or sofware quality attributes
Extraversion	Decomposition, modularization, testability, functionality, reusability, programming style	[AGJ09]	
Extraversion, Conscientiousness	High performance	[SMGS09]	
Extraversion, Openness to Experience	Open Source development	[RH07]	
Openness to Experience	Pair programming effectiveness	[SMGS10]	
Intuitive and Thinking	Code review	[DG07]	
Experience	Issue lead time/Bug fixing time prediction	[ACCA11]	Experience
Experience	Interpretation of user's needs, requirements elicitation	[Kai85]	
Experience, reflection, motivation, personal characteristics (thoroughness, conscientiousness, patience or persistency, accurateness, creativeness), domain knowledge, specialized technical skills	High performing tester	[IMI10]	

cont.: Summary of influencing factors on software quality or performance.

Distinct personality traits and their impact on performance or software quality attributes

This	*has impact on*	*reports who*	*group*
Self-efficacy	In major grade point average of computing students	[BLK+07]	Self-efficacy and values
Personality (high selfesteem, high selfefficacy, high locus of control, low neuroticism), cognitive ability, value belief	Object Oriented programming performance	[CH06]	
Values	High competence	[GB10]	
Short term memory performance	Location of high frequencies of errors; quality of a software system	[RK03]	Cognitive abilities
Cognitive weight of a program	Software complexity and physical size of program	[JY03]	
Student's perception of understanding of the module	Programming performance	[SR05]	Miscellaneous
Human factors, individuality	Design cost and quality including design time, design efficiency, and designer error rate	[AS95]	
Communication, domain knowledge	Requirements elicitation	[MPP11; YR08]	
Skilled people	High quality software	[Miz83]	
Openness to Experience	High job performance	[Gal04]	**No link** between personality (traits) and programming performance
Personality	Excellent programmer	Not enough research done to prove link; [DM05]	

analyzed experts as study subjects and not students. The most often used personality type test or theory was the Myers-Briggs-Type-Indicator. For RQ2, studies predominantly analyzed students. If there was used a personality type test for describing personal attributes, researchers often chose the Big Five Theory. Figures 3.6 and 3.7 illustrate these results.

Some papers could not be associated to any of the former mentioned groups. These papers will be described in the following.

Christiansen et al. investigated "the effects of congruence between personality and task-demands on job satisfaction" [CSF14]. They found that "tasks associated with Agreeableness and Conscientiousness were perceived as more distressing when workers were low on those traits, and increased distress was related to less satisfaction across task domains. ... individuals high in Neuroticism tended to evaluate all tasks as being more distressing and most notably those related to Extraversion." [CSF14] This also supports our theory that the task has to fit the personality type and supports our presumption that personality type influences the outcomes of the work - if work is found to be very stressful and not satisfactory, the quality of the work (outcome) is most likely negatively influenced.

Soomro et al. examined in literature "What are the key factors in team climate composition and which team climate factor(s) has effects on software team productivity" [SS14]. They found that "team climate factors such as collaboration, cooperation, coordination, collective thinking and cohesion are synonyms to each other and therefore they have significant impact on software team performance." [SS14] Other factors they found were "role allocation to a team member in software development team and participative safety, which is one of the inventories of TCI (Team Climate Inventory) ... . Both of these factors were found significant in influencing software development teams' performance." [SS14]. The issue of role allocation will be resumed in Chapter 5.

Bishop and Deokar investigated which personality factors influence the "preference for the agile software development approaches" [BD14]. They found that extraversion and openness to experience "have a positive relationship with agile preference while neuroticism (emotional instability) has a negative relationship with agile methodology preference" [BD14]. This supports our findings reported in Chapter 5. Agile development methods are common nowadays [Den16; Jer]. Combined with the findings of Soomro et al. [SS14] if personality affects preference for agile development methods then personality factors also influence the team climate and thus team productivity. This leads to the assumption that personality factors also have an influence on software quality. Chapter 4 reports of our findings concerning correlations between personality type and maintainability of software.

Two researchers conducted a systematic literature review on a very similar topic. They did not focus on the impact of personality on software quality but "discuss(ed) indicators for a relationship between task-personality fit and software engineering performance" [WK14].

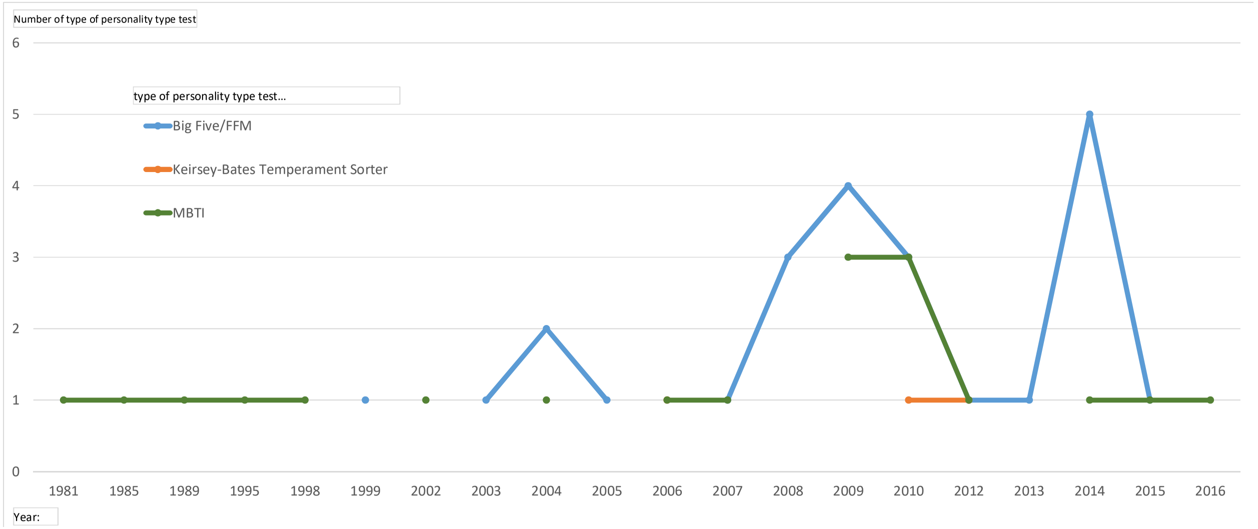


Figure 3.1.: Number of studies using different personality type tests grouped by year and personality type test. This figure only contains the 3 mostly used personality type theories which we found in our study.

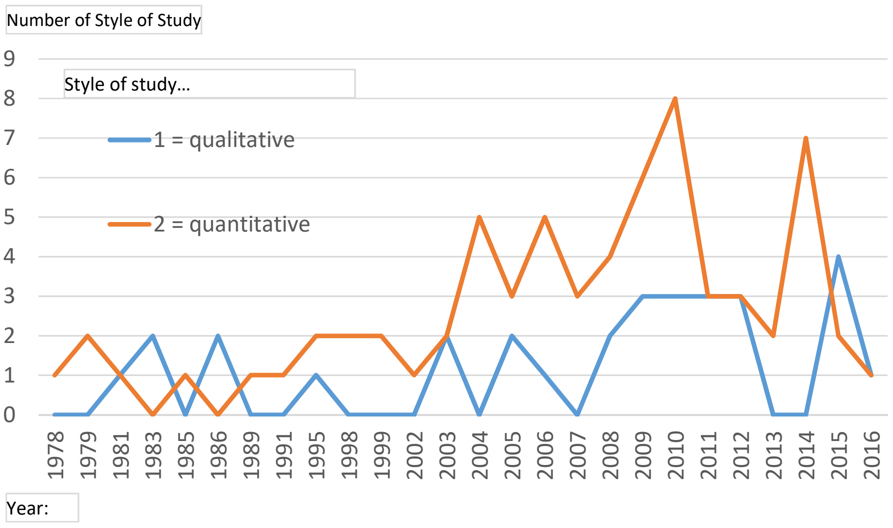


Figure 3.2.: Number of studies incorporating personality clustered by qualitative and quantitative studies and grouped by year.

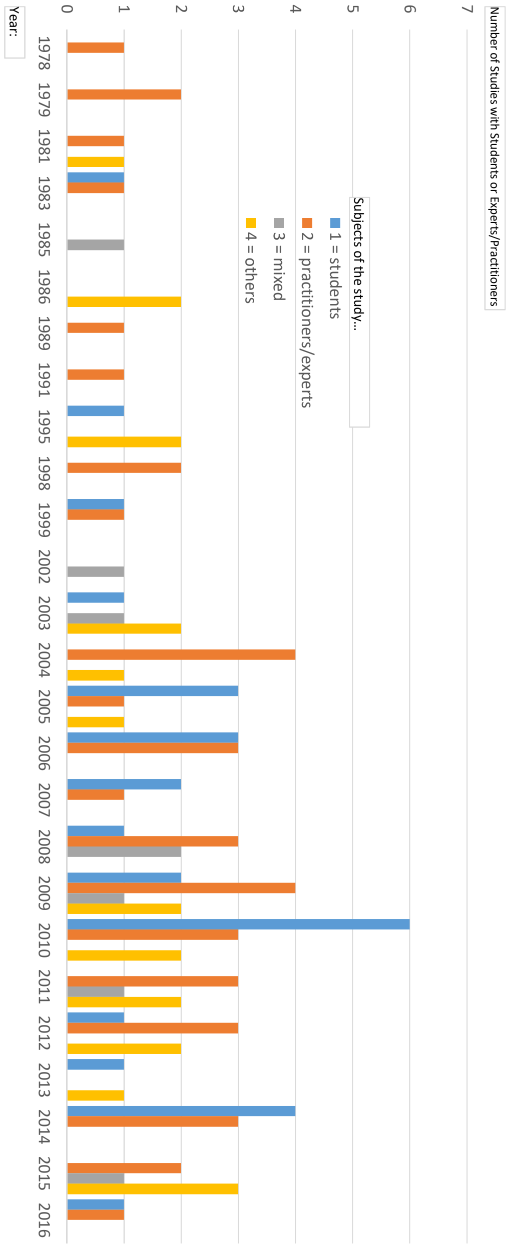


Figure 3.3.: Number of studies using students or experts grouped by year.

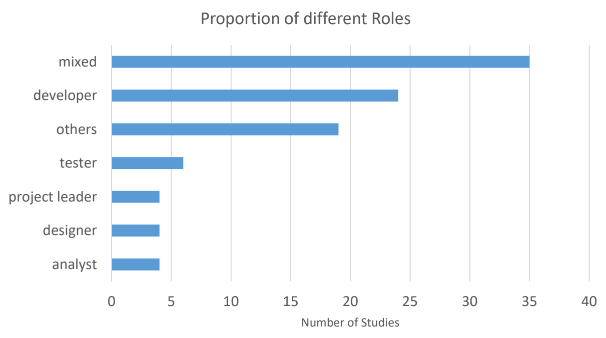
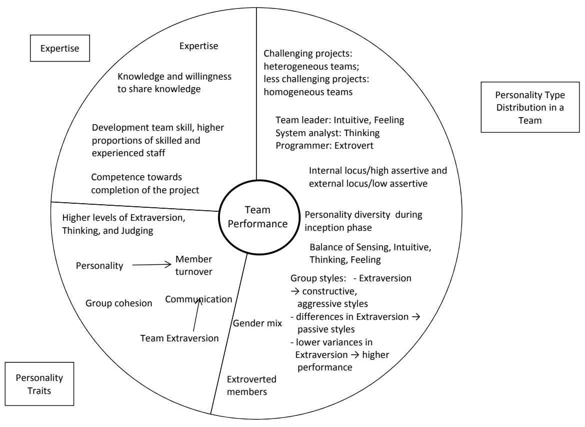


Figure 3.4.: Proportion of the different roles reported in studies.



Expertise
Expertise
Knowledge and willingness to share knowledge
Development team skill, higher proportions of skilled and experienced staff
Competence towards completion of the project
Challenging projects: heterogeneous teams; less challenging projects: homogeneous teams
Personality Type Distribution in a Team
Team leader: Intuitive, Feeling
System analyst: Thinking
Programmer: Extrovert
Internal locus/high assertive and external locus/low assertive
Team Performance
Personality diversity during inception phase
Balance of Sensing, Intuitive, Thinking, Feeling
Higher levels of Extraversion, Thinking, and Judging
Personality
Member turnover
Group cohesion
Communication
Team Extraversion
Gender mix
Group styles: - Extraversion → constructive, aggressive styles
- differences in Extraversion → passive styles
- lower variances in Extraversion → higher performance
Extroverted members
Personality Traits

Figure 3.5.: Three categories covering research question 2.1.

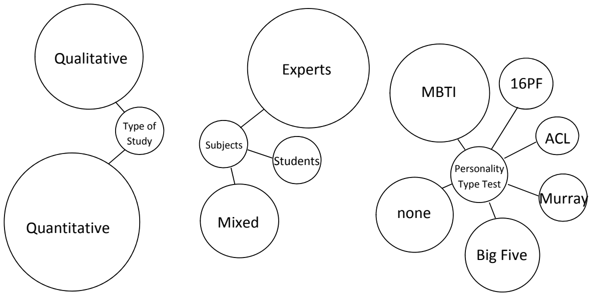


Figure 3.6.: Rough distribution of the number of studies concerning RQ1 arranged by type of study, study subjects, and personality type test.

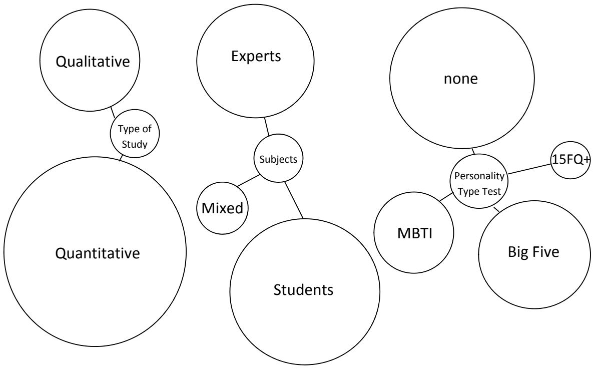


Figure 3.7.: Rough distribution of the number of studies concerning RQ2 arranged by type of study, study subjects, and personality type test.

3.5. Discussion

Many researchers investigated the influence of human factors or personality on performance and even software quality in general. Although it is a very interesting and important issue, the question of how the personality of a software engineer influences distinct software quality aspects has so far not yet been studied to a satisfying level. We could detect only two studies which explored this link AND named the investigated software quality attributes and measures. The authors of the other studies did not mention any reason for this imprecise information. Quite the opposite, we had the impression that very often quality was evaluated intuitively. This might be due to the reason that it is still a quite unsolved problem to measure software quality objectively and especially to compare the software quality of different software artifacts objectively.

In addition, many studies used the Myers-Briggs-Type-Indicator whose reliability and validity has been questioned [HLW03]. Most studies used student groups as study subjects. This makes it difficult to generalize the results to professionals, since personality changes with major life events like the first job [Ahl18; SES11].

3.6. Conclusions

Literature shows that there are connections between the personality of a software engineer and performance or software quality in general. But research done so far is not sufficient to provide a strong basis for making clear statements on the link between the personality of a software engineer and distinct software quality attributes. With this literature review we have shown the necessity of further studies in this area. Studies should include larger groups and study subjects should be experts. It is also important to chose a personality type test whose reliability and validity has been proven.

Further studies should be conducted among experts from different branches of industry. Studies could concentrate on particular roles of people in a software engineering process, e.g. architects, programmers, or testers. As mentioned above, a valid and reliable personality type test should be used for the investigations. Additionally, studies could concentrate on one aspect of software product quality and should clearly define the metrics that were used to measure the distinct aspect of software product quality. Having all these data, researchers could investigate all kinds of correlations between personality type and different aspects of software product quality.

Motivated by the results of this literature review, the next Chapter presents a concrete examination of the relationship between personality and software quality.

Chapter 4

Tell me Who You Are and I Will Tell You How Your Code Looks Like

About the Prediction of Maintainability by Personality

The Literature Review has shown that there are not yet enough quantitative studies to make a reliable statement about the relationship between software quality and the personality of a developer. Furthermore, software quality was often not precisely defined in existing studies or subjective measures were used.

In this chapter a quantitative study is presented, which examines the connection between software quality and the personality of the developer. Since "software quality" covers a large range, this investigation concentrates on one aspect of software quality – maintainability.

Maintenance takes up a huge part of software costs. Many models have been developed to predict software maintainability. And many metrics have been defined to assess maintainability. But they put their focus on technical factors and do not take into account human factors of software developers, e.g. their personality. It has been proven, that personality influences the way a person writes a text. So it is legitimate to suggest that personality influences the way a person writes code. One aim of this chapter is to develop a model which enables a prediction of maintainability aspects by the prevalent personality trait of the software engineer who developed the software. With our research findings, software developing companies could reduce their maintenance costs, e.g. by optimal staffing or supporting their staff with tailor-made trainings, and increase the satisfaction and motivation of their employees. Another aim is to demonstrate the applicability and advantages of the Bayesian data analysis method for this type of research – we explored differences between several groups with different personality traits. By this we also provide a basis for further analyses in this area because our data can be used as priors for further Bayesian data analyses.

4.1. Motivation

Typically, maintenance makes 40%-80% of the software costs [Gla01]. So would it not be an advantage for companies to enhance maintainability right from the start of the development process?

In the past, many models for predicting software maintainability have been developed. Metrics have been defined to assess maintainability. But almost all of them focus on the software itself or the development process. And hardly any of them take into account the people – with their variety of personalities – who are directly involved in the development process and thus have a direct influence on the software and its respective quality attributes.

Researchers who investigated "the extent to which compatibility of personality influences online interaction" found out that the study subjects reading a "blog corpus were able to consistently judge the personality of the writers based solely on the text that they wrote." [LC10]. In other words: Personality influences the way a person writes a text. These results suggest the assumption that the way of writing software – and thus associated software product quality attributes – is influenced by personality, too.

In this Chapter, we explore the link between the prevalent personality trait of a software engineer and the maintainability of the software, she or he developed. The goal is to develop a model which describes this link. Results of this research are valuable for any software developing company. The companies will be supported in their selection of personnel as well as in the choice of further training to support their staff in order to increase the output of high quality software with respect to maintainability factors. The benefit is twofold: Firstly, the maintainability of the software will be enhanced which can be measured by money. And secondly, the motivation of the employees will be increased because they are more pleased with the results they produce.

4.2. Related Work

In order to investigate relationships between personality traits and software quality, we conducted a systematic literature review (see Chapter 3). Since the literature review is very in-depth, we do not expand the Related Work section in this Chapter, but draw on the results of the literature review.

Results of this systematic literature review indicate that some small studies already proved the influence of personality on various software quality attributes like ([AGJ09], [DM05], [IMI10], [OS10], [PKS06]). The literature review also identified the following gaps:

- Far too few investigations have been made in the industrial environment, most study subjects were students.
- The number of studied subjects was too small to make statistically significant statements.
- The personality (type) tests used were not valid and reliable.

This study was be different in the following ways:

- The study subjects were programmers from an industrial environment. AND
- We used a personality test which is reliable and valid.

We investigate the relation between personality traits and maintainability of the software. Understanding code makes a huge part in maintenance (see Chapter 2.2). Thus the way source code is written is a very important factor for the maintainability of the source code. Programming style is one influencing factor of maintainability

[BSB08, p. 26]. "Maintenance begins with the development. Thus maintainability is crucial shaped during the first development" [BSB08, p. 26]. Of course, the keystones of programming – syntax and semantics as well as programming paradigms, programming standards and comparable things – can be acquired. Nevertheless, the implementation of the aforementioned depends on the programmer him- or herself. Therefore personal aspects play a significant role when we talk about software quality (for some ideas see e.g. [GZG+14; Nei08; Vas14]). Because of this, it would be beneficial to know in which way personality influences the maintainability of software so that developers can be specifically and individually trained to pay attention to specific aspects of software maintainability.

4.3. Study Design and Execution

Our study is of observational character because the nominal predictor (personality traits) as well as the metric predicted value (software maintainability metrics) "are generated by processes outside the direct control of us" [Kru15, p.553f]. We could only make specifications for the time of the measurements.

Goals
Our goal is to describe the relationship between the prevalent personality trait of a software developer and the maintainability of the software, she or he developed. Ideally we can form a predictive model.

Hypotheses
We assumed that the prevalent personality trait of a software developer has an influence on the quality of the code, in particular on the maintainability of the code. Conscientious developers e.g. might write more lines of code but maybe also more documentation. Developers with high scores on the Aesthetic Appreciation scale might try new ways of structuring code and therefore the code might become more understandable. Or on the contrary developers with high scores on the Anxiety facet level feel stressed by bigger changes and might tend to stick to the old or become unstructured. Thus we were interested in predicting maintainability based on the personality of the developer.

Parameters, Objects and Instrumentation
One parameter or variable we needed were the software quality metrics we defined in Chapter 2.2. We have therefore provided the participating companies with the

VizzMaintenance[1] tool as we used it in an academic context. The reason for providing the companies with this tool was to ensure a uniform measurement of the metrics. If we had used different tools, the metrics could have been defined differently.

In order to make the data comparable, we have set some boundary conditions for the participants and the measured projects as well as the measurement times. Participating developers should have made at least 20 commits to the project and the project must not have been worked on for longer than 2-3 years. There were no restrictions regarding the size of the project. Any number of commits could lie between the two measurement points if they came from the same developer because it is possible that only minimal changes have been made in a commit. Developers should select commits where they have made real changes (feature included, technical debits removed, etc.). There was no restriction concerning the size of the code (e.g. mandatory minimum of LOC). And there was no specified time span between the two commits. Measured code should only be written in Java. One company contact of each company sent us the collected measurement results via e-mail. Measurement results included two datasets of each participating developer – one dataset contained the metrics before a commit and the other dataset contained the metrics after the aforementioned commit.

We used the HEXACO personality inventory revised (HEXACO-PI-R) [Ash18] in order to describe personality traits. For measuring personality traits we used the German 60 questions version of the HEXACO-PI-R taken from [LA09a]. We provided each company contact with a URL to an online version of this inventory. The company contacts distributed this link to the developers. For the online version of the personality inventory we used the tool LimeSurvey[2].

In order to link the anonymous personality inventory to the anonymous metrics measurement results, each participant generated a personal code according to a predetermined pattern and wrote this code in a special field in the personality inventory and also named the measurement files with this code. Thus anonymity was granted also within the companies as pattern was constructed in a way the code could not be reconstructed by others.

Variables

As required first step in Bayesian analysis, we "identify the data relevant to the research question" [Kru15].

[1]http://www.arisa.se/products.php
[2]https://www.limesurvey.org/de/

To be predicted (i.e. **dependent**) variables are the maintainability metrics:

- Coupling Between Objects (CBO)
- Data Abstraction Coupling (DAC)
- Depth of Inheritance Tree (DIT)
- Improvement of Lack of COhesion among Methods (ILCOM)
- Lack of COhesion among Methods (LCOM)
- Lack Of Documentation Class (LOD_Class)
- Length of Names (LEN)
- Lines of Code (LOC)
- Locality of Data (LD)
- Message Pass Coupling (MPC)
- Number of Attributes and Methods (NAM)
- Number Of Children (NOC)
- Number of Classes in Cycle (Strongly Connected Component) (CYC_Classes)
- Number Of Methods (NOM)
- Response For Class (RFC)
- Tight Class Cohesion (TCC)
- Weighted Method Count (WMC)

They are of the form as shown in Table 4.1.

We chose the number sets $\mathbb{Z}$ and $\mathbb{R}$ because for computing the values of the metrics, we calculated the **difference** of the values each developer caused in the software. Reason for this is that software is developed in teams and if we just would have measured the maintainability metrics after a commit of a developer, we would have also included all the code which was added by the other developers in the team. Thus we calculated the difference between two commits of one developer with the specification that no other developer has commited code between those two points in time. We did not limit the number of commits of the single developer which could have taken place between those two points. But no foreign developer was allowed to commit in between. Thus also negative numbers could appear, e.g. if a developer deleted a class or lines in the code.

The ratio scale is based on the fact that there is a zero point for each of the metrics collected.

Table 4.1.: Predicted (dependent) variables

CBO	metric with ratio scale	$\mathbb{Z}$
DAC	metric with ratio scale	$\mathbb{Z}$
DIT	metric with ratio scale	$\mathbb{Z}$
ILCOM	metric with ratio scale	$\mathbb{Z}$
LCOM	metric with ratio scale	$\mathbb{Z}$
LOD_Class	metric with ratio scale	$\mathbb{R}$
LEN	metric with ratio scale	$\mathbb{Z}$
LOC	metric with ratio scale	$\mathbb{Z}$
LD	metric with ratio scale	$\mathbb{R}$
MPC	metric with ratio scale	$\mathbb{Z}$
NAM	metric with ratio scale	$\mathbb{Z}$
NOC	metric with ratio scale	$\mathbb{Z}$
CYC_Classes	metric with ratio scale	$\mathbb{Z}$
NOM	metric with ratio scale	$\mathbb{Z}$
RFC	metric with ratio scale	$\mathbb{Z}$
TCC	metric with ratio scale	$\mathbb{R}$
WMC	metric with ratio scale	$\mathbb{Z}$

The predictor (i.e. **independent**) variable is personality with the the personality traits as categories or levels:

- Level 1: Honesty-Humility
- Level 2: Emotionality
- Level 3: eXtraversion
- Level 4: Agreeableness (versus Anger)
- Level 5: Conscientiousness
- Level 6: Openness to Experience

Our predictor is nominal (non-metric) with a nominal scale (the six different personality traits).

This means we have metric predicted variables with a non-metric predictor.

4.3.1. Sample and Data Collection Procedure

For reasons of comparability and due to the fact that Java is the second most common and the most popular programming language in the world [H17; Moh18; Ric18] we investigated only Java code and therefore, our focus was on companies that develop software in Java. From November 2015 till June 2019 we contacted 28 companies in Germany covering a broad field of branches starting from automotive over finance and insurance to companies that produce individual software of any kind. Only 3 of these companies agreed to take part in our study.

There were plenty of reasons why the companies were hindered. One reason was the collection of personal data with the personality inventory. Although we ensured anonymity, this test has to be approved by the works council. For some companies this was too much of an effort. Another reason was that some companies guarantee a strict confidentiality of the software to the customer. This also included software quality metrics of any kind. This means that although we didn't have to see the code in our investigation, those companies still couldn't provide us with the metrics.

Parallel, we tried another attempt in collecting data of github open source projects. For this we selected Java projects, which were currently still being worked on. The first two projects we contacted were commercial projects. Since we did not want to send "spam mails", we first asked the project managers if we could send our survey to the project members. We contacted two companies. The project managers of both companies agreed that we could send a mail to the team members. From one company only 4 of 41 contacted developers took part in the survey.

Before we could contact the project members of the second company, the GDPR[1] "got in our way", already in 2016/17. All e-mail addresses in github were stored encrypted and the project leaders did not tell us the "plain" e-mail addresses of the project members. Thus it was not possible to collect data from the second open source project. So we discarded this path.

In the end, data of two companies were analyzable because one of the three companies provided us only with the software metrics but not with the personality inventory results, although we kindly reminded them for three times.

4.4. Analysis

4.4.1. Descriptive Statistics

The two participating companies were from the finance and insurance field and a manufacturer of individual software. We had 10 participants who filled out the

[1]General Data Protection Regulation, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN

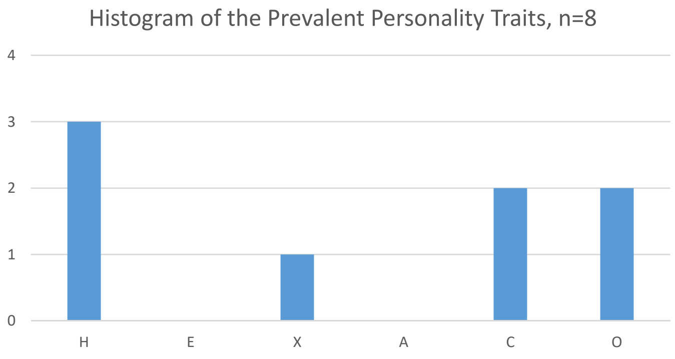


Figure 4.1.: Histogram of the prevalent perosnality traits, $n = 8$

personality inventory and sent us the results of their metrics measurement. The metrics of two of them did not help us because the difference of both measured points in time was zero. Thus our final number of participants was 8.

We evaluated the personality inventories according to [LA09a]. This inventory contains 10 questions for each of the 6 dimensions with an equal distribution of points. We summed up the points for each dimension according to the instructions given in [LA09a]. We then searched for the maximum value of the 6 sums and define this dimension as "prevalent personality trait". We did not compare the sums of each dimension to averages taken from the population. Our intention was to compare the different characteristics of the participants themselves with each other.In our analysis we integrated only those participants who completely filled out the personality inventory and and whose results in the measurement of metrics were not equal to zero. Thus the final number of participants is 8. Figure 4.1 shows a histogram of the distribution of the different prevalent personality traits.

Two of the participants were female. On average all participants have 13,875 years of professional experience, the median is 15 years. Other information like current job title or age you find in Figure 4.2.

4.4.2. Bayesian Statistical Inference

As our sample consisted of only 8 participants, we do not meet the assumption of normality [Fie13, p.168] which has to be met for doing regression analysis. Also the

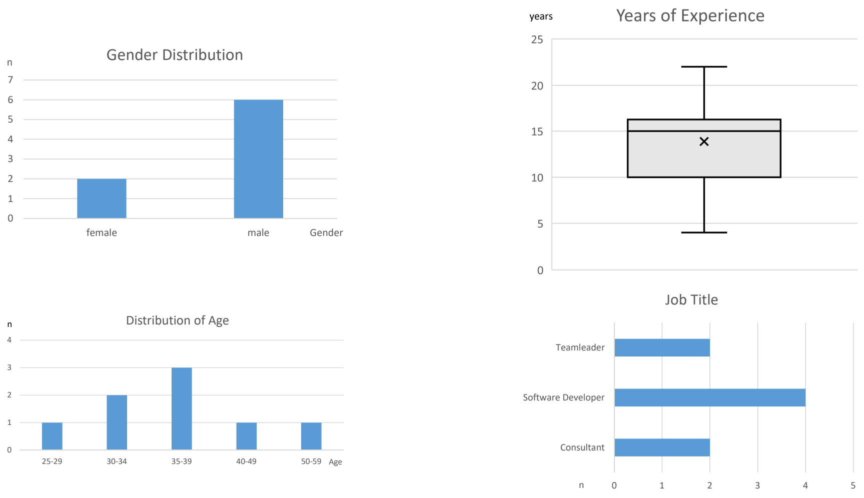


Figure 4.2.: Additional information about the participants, $n = 8$

number of datapoints each individual contributed to the data set varies. And we were interested in more than "only" acceptance and rejection of hypothesis – we wanted to learn about differences between groups with different prevalent personality traits and even quantify them. Therefore, we have not performed "classical" statistical tests – in the sense of frequentist statistics – to test our assumptions but have chosen a Bayesian approach [Kru15]. This means we conducted Bayesian statistical inference and not null hypothesis significance testing.

We have to keep in mind that " "prediction" does not imply causation" [Kru15, p.421]. With our model we can predict the maintainablity from the personality but this does not neccessarily mean that personality *causes* those maintainability outcomes. For the analysis of causality, much more data points are needed so that causal models can be created and structural equation analyses can be performed.

Bayesian data analysis roughly consists of these three steps [GCS+14]:

1) "Setting up a *full probability model*– a joint probability distribution for all observable and unobservable quantities in a problem."
2) "Conditioning on observed data: calculating and interpreting the appropriate *posterior distribution*"
3) "Evaluating the fit of the model and the implications of the resulting posterior

distribution”

4.4.3. Setting up a probability model

We do not have a strongly informed prior, e.g. from literature or the initial scatterplots of the data. And as our dataset is not large enough to make strong corrections to an incorrect prior, we chose a "broad and noncommittal distribution for the prior" [Kru15, p.114].

"Despite the fact that many real-world dependencies are nonlinear, most are at least approximately linear over moderate ranges of the variables." [Kru15, p.424]. Therefore we chose a linear model to describe our data.

We have one predictor - personality. This predictor has six different categories - the six personality traits. The predictor is represented as vector $\vec{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$. When a person has the prevalent personality trait j, this is represented by setting $x_j = 1$ and $x_{i \neq j} = 0$. For example if a person's prevalent personality trait is Honesty-Humility, this is represented by the vector $\vec{x} = (1, 0, 0, 0, 0, 0)$ as Honesty-Humility is Level 1 (see 4.3).

The predicted value y therefore is

$$y = \beta_0 + \vec{\beta} \cdot \vec{x} \tag{4.1}$$

with the constraint

$$\sum_{j=1}^{6} \beta_j = 0 \tag{4.2}$$

where β_0 is the baseline and $\beta_j, j = 1, .., 6$ are the deflections from the baseline which have to sum up to zero [Kru15, p.430].

As link function [Kru15, p.435] we chose the identity function because in our case, values of the metrics range from negative infinity to positive infinity (remember that our metrics values express the **difference** of metrics between two points in time). And we assume a linear connection between personality traits and metrics. Thus the link function $f()$ is

$$f(lin(x)) = lin(x) \tag{4.3}$$

where $lin(x)$ is a linear function (see [Kru15, p.443]).

"We predict that y tends to be *near* $f(lin(x))$" [Kru15, p.441]. Therefore we define $\mu = f(lin(x))$ where "μ represents the central tendency of the predicted y values" [Kru15, p.441]. As our predicted values are of metric scale type, we chose the normal distribution as typical noise distribution of such values [Kru15, p.443] and note

$$y \sim normal(\mu, \sigma) \tag{4.4}$$

with

$$\mu = lin(x) \tag{4.5}$$

The inverse link funktion $f()$ in 4.3 and μ in 4.5 toghether with the distribution of the predicted variables 4.4 form the "General Linear Model" [Kru15, p.444].

4.4.4. Calculating and Interpreting the Posterior Distribution

For data analysis we used the system JAGS 4.3.0[1]. "JAGS takes a user's description of a hierarchical model for data, and returns an MCMC sample of the posterior destribution." [Kru15, p.194].

As we have one nominal predictor and our dependent variables are metric, we chose the "Jags-Ymet-Xnom1fac-MnormalHom.R" model by Kruschke [Kru15, p.553ff]. The y variable were the metrics, the x variable was the personality code. In our data, the personality inventory was evaluated and each participant's personality code was the abbreviation of the prevalent personality trait, e.g. if a participant scored highest in Extraversion, X was the associated personality code. The list of codes is:

H: Honesty-Humility
E: Emotionality
X: eXtraversion
A: Agreeableness
C: Conscientiousness
O: Openness to Experience

We ran the model for 17 times, one time for each single metric. You can find the simulated data with the posterior predictive distributions in Figure 4.3. The x axis

[1]Just Another Gibbs Sampler, http://mcmc-jags.sourceforge.net/

Table 4.2.: Resulting values for β_0 and the vectors $\vec{\beta}$ for each metric

Metric	β_0	$\vec{\beta}$
CBO	0,97	(0,11; 0; 0,59; 0; -0,36; -0,34)
CYC_Classes	0,74	(0,46; 0; 0,10; 0; -0,32; -0,24)
DAC	0,91	(0,17; 0; 0,62; 0; -0,42; -0,38)
DIT	0,08	(0,05; 0; -0,08; 0; -0,06; 0,09)
ILCOM	0,64	(0,24; 0; 0,28; 0; -0,47; -0,05)
LCOM	27,25	(14,88; 0; -11,64; 0; 5,41; -8,65)
LD	0,18	(-0,08; 0; 0,27; 0; -0,13; -0,05)
LEN	8,11	(0,21; 0; 4,66; 0; 0,98; 2,25)
LOC	34,45	(19,15; 0; 9,54; 0; -20,55; -8,14)
LOD_Class	0,24	(-0,13; 0; 0,34; 0; -0,16; -0,04)
MPC	3,32	(2,90; 0; -1,26; 0; 0,30; -1,95)
NAM	4,13	(1,63; 0; 1,89; 0; -1,72; -1,80)
NOC	0,10	(0,11; 0; -0,08; 0; -0,03; 0,008)
NOM	2,52	(0,41; 0; 1,29; 0; -1,01; -0,69)
RFC	3,62	(0,46; 0; 1,45; 0; -0,93; -0,98)
TCC	0,10	(-0,03; 0; 0,14; 0; -0,05; -0,06)
WMC	2,93	(0,54; 0; 1,48; 0; -1,36; -0,67)

shows the prevalent personality trait, the y axis shows the different metrics. The figures all together represent the model which predicts the single metrics from the HEXACO personality traits.

The resulting factors β_0 and the vectors $\vec{\beta}$ to describe the linear model 4.1 are given in Table 4.2.

Please keep in mind that this section only presents the analyzed data, you will find an interpretation in the next section.

We have also examined the differences between the personality trait groups for each metric. You find plots for the difference between H-X, H-C, H-O, X-C, X-O, and C-O for each metric in Figure 4.4.

4.4.5. Evaluating the Fit of the Model

In Fig. 4.3 the data are plotted as red circles in each figure. The curves show that our assumptions about the normal distributions with homogeneous variances seem to be an adequate description of our data. We see only some outliers relative to the

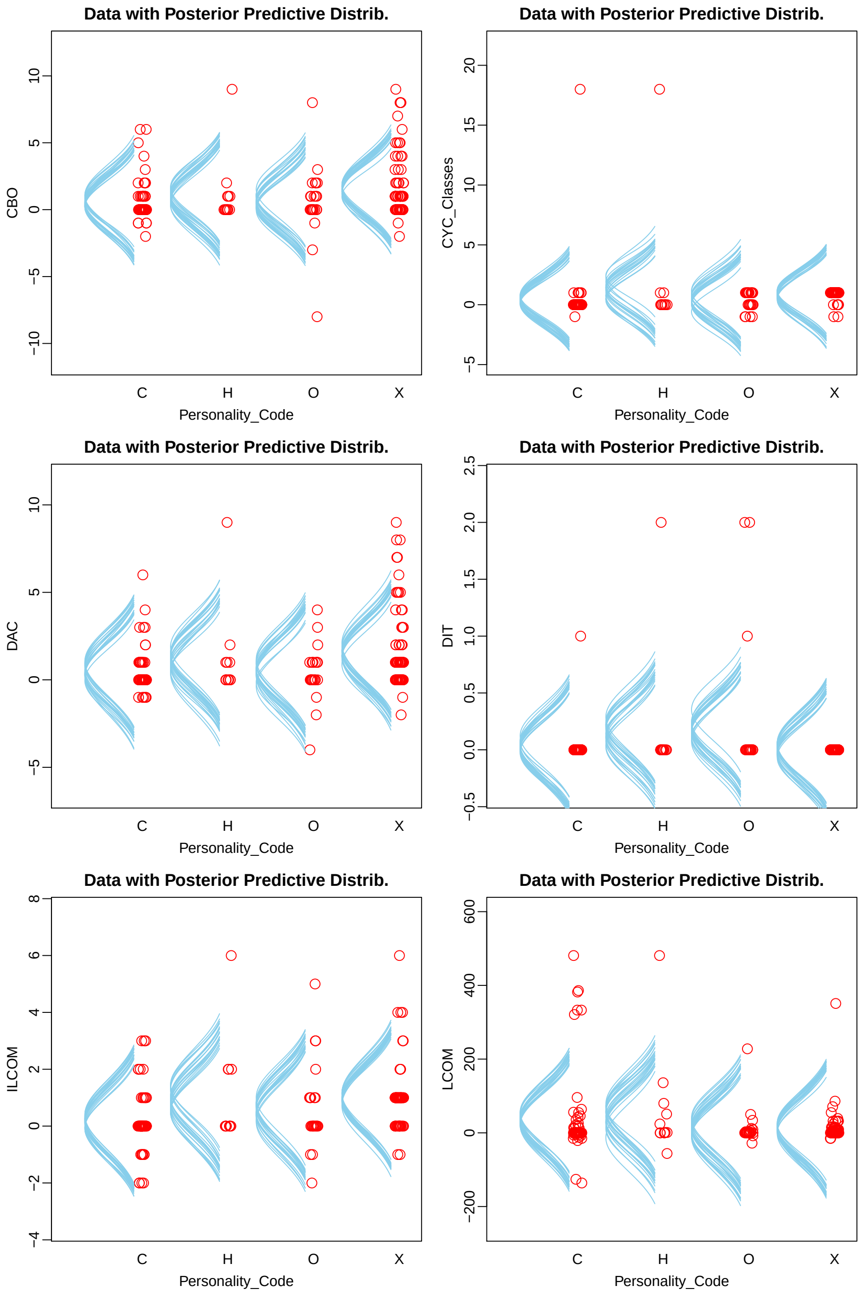
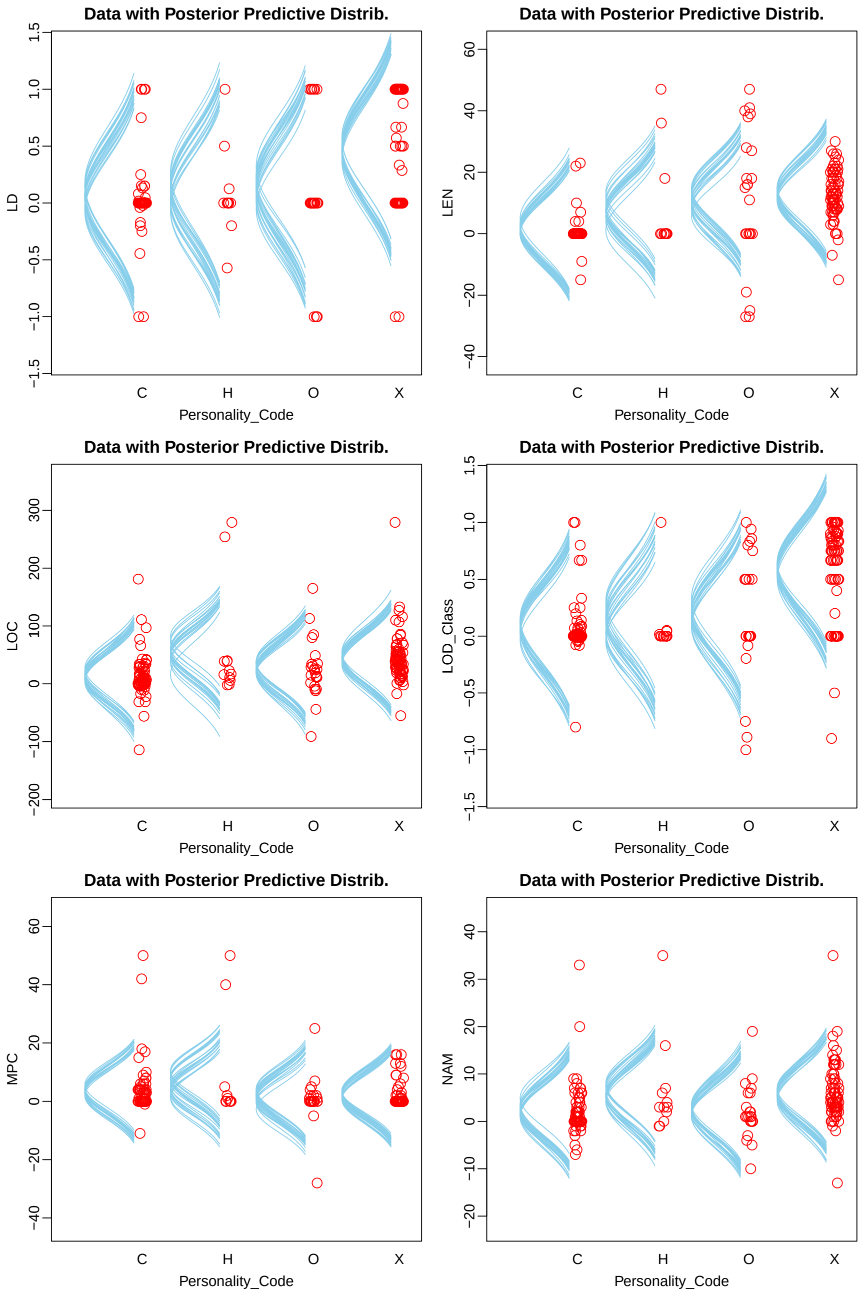
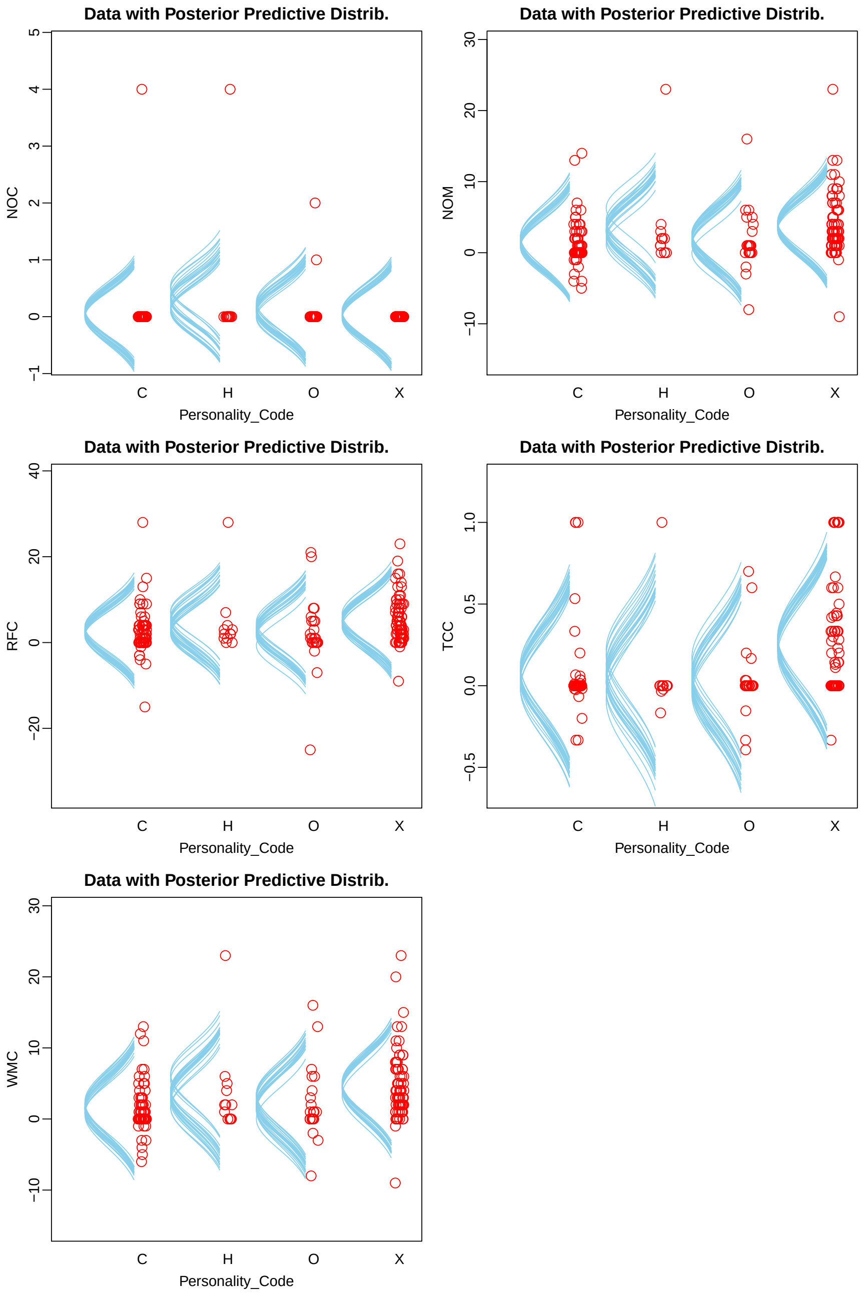


Figure 4.3.: Posterior predictive distributions for each metric



Posterior predictive distributions for each metric



Posterior predictive distributions for each metric

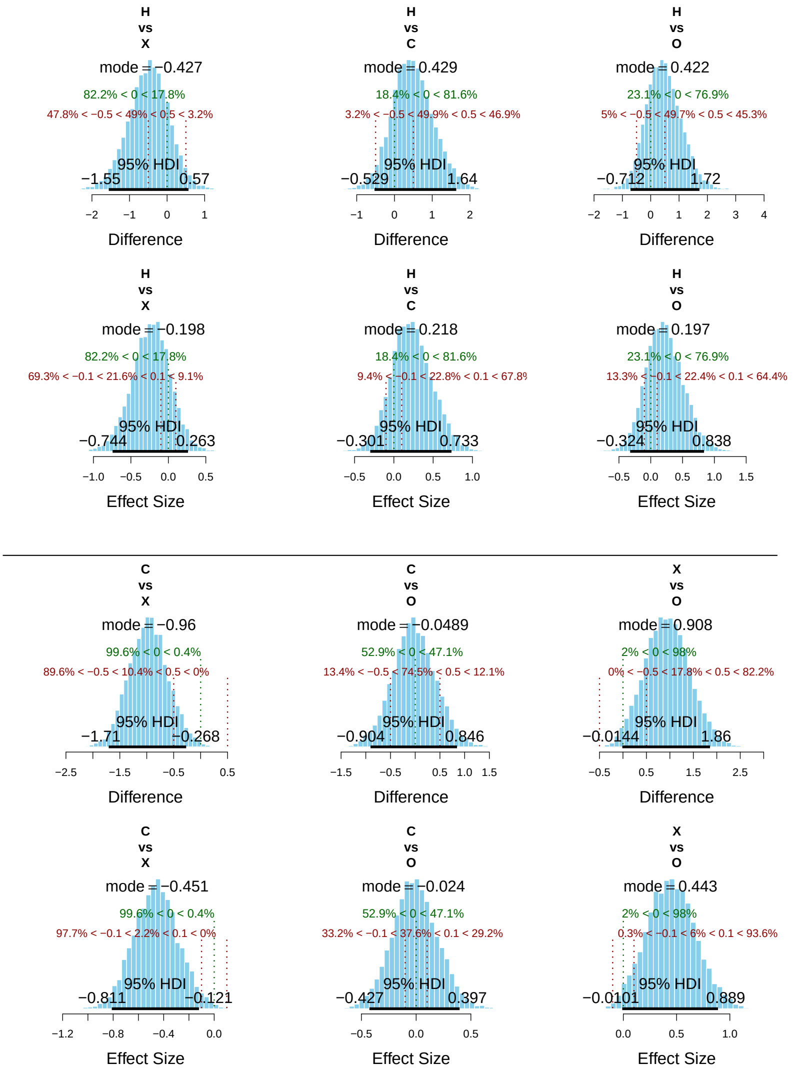
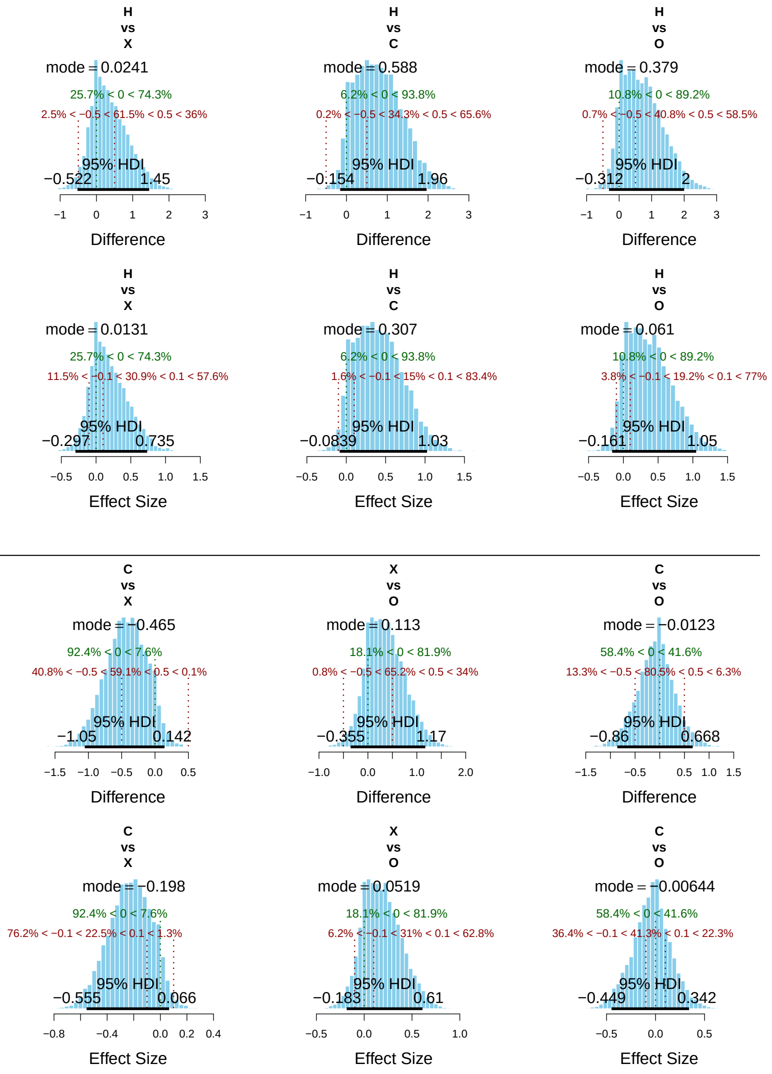
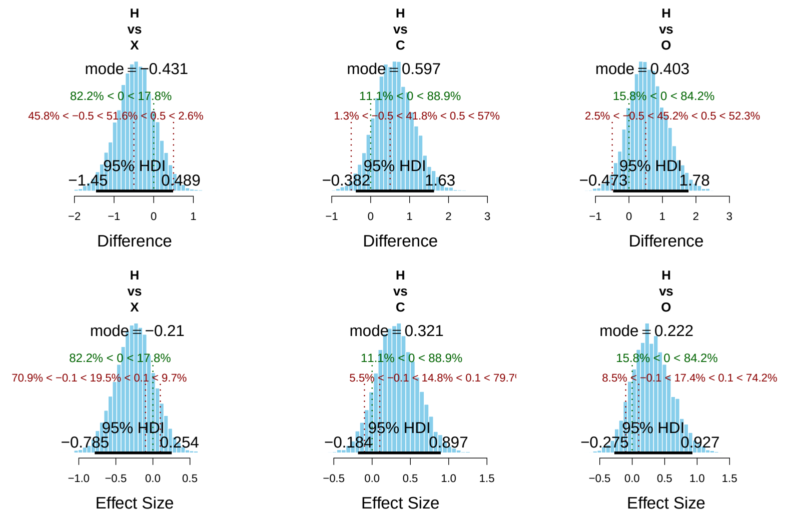
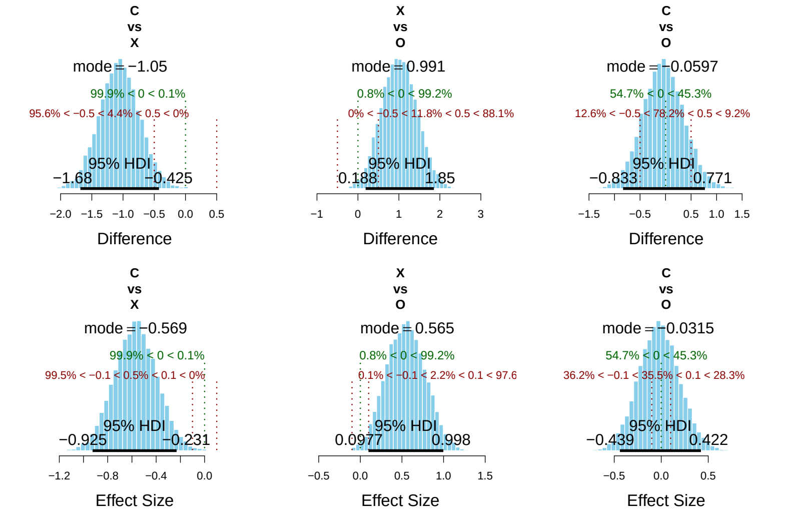


Figure 4.4.: Distribution of differences between groups for metric CBO

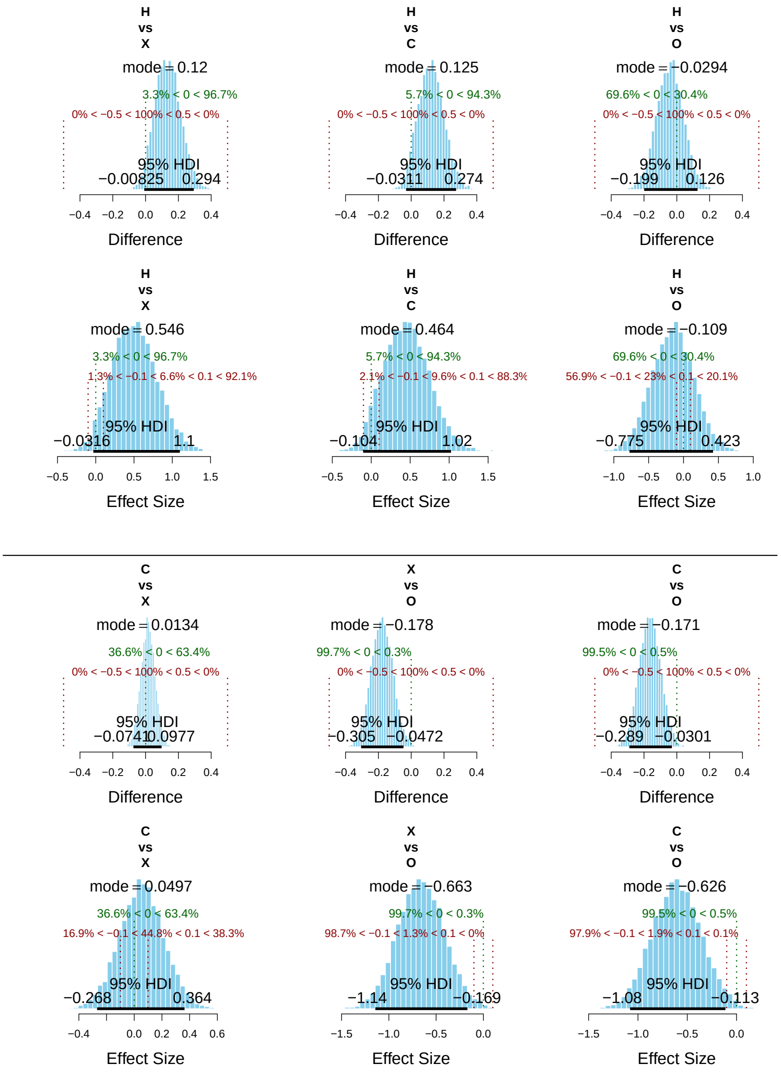


Distribution of differences between groups for metric CYC_Classes

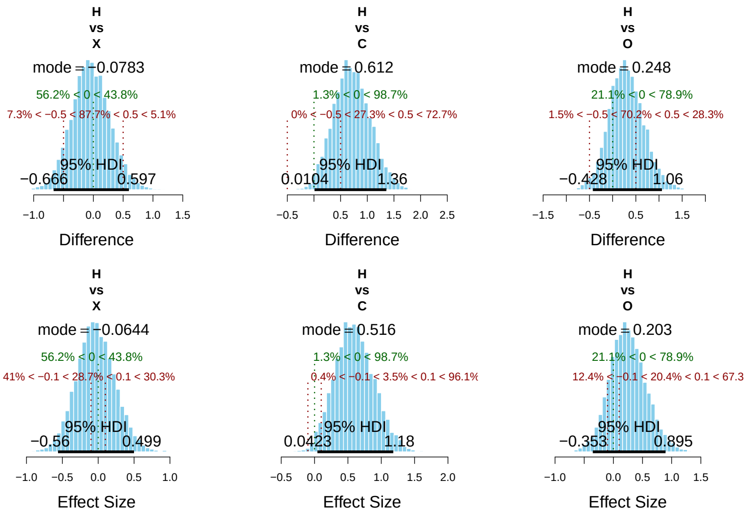


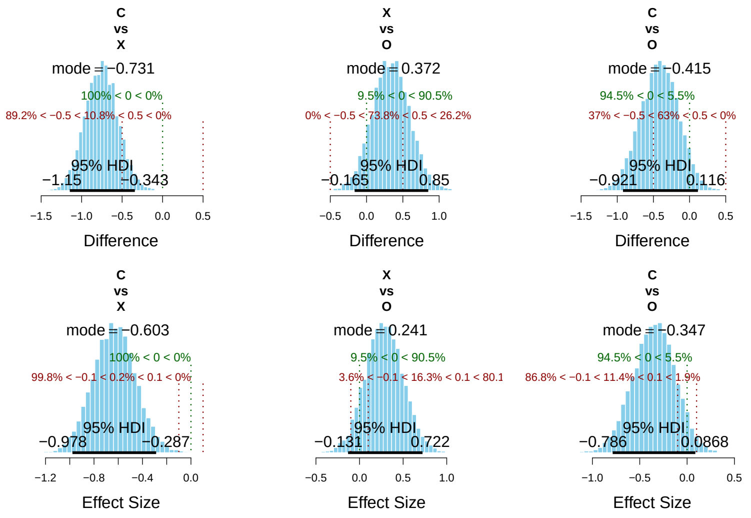


Distribution of differences between groups for metric DAC

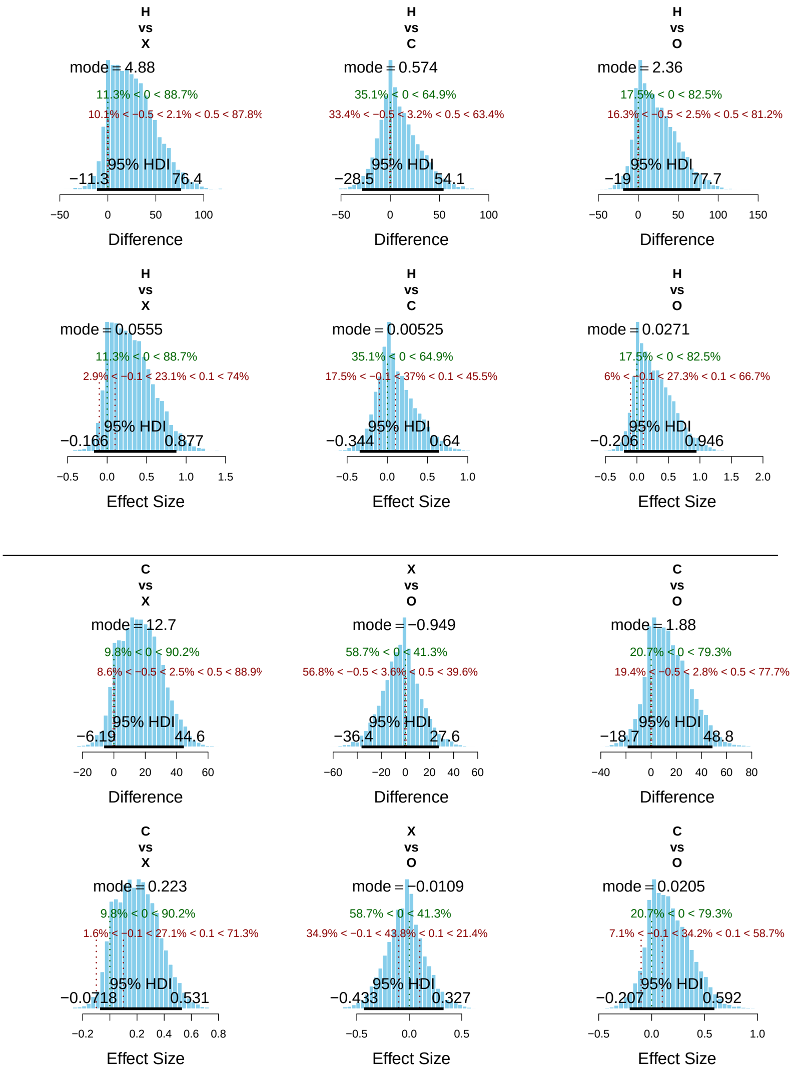


Distribution of differences between groups for metric DIT

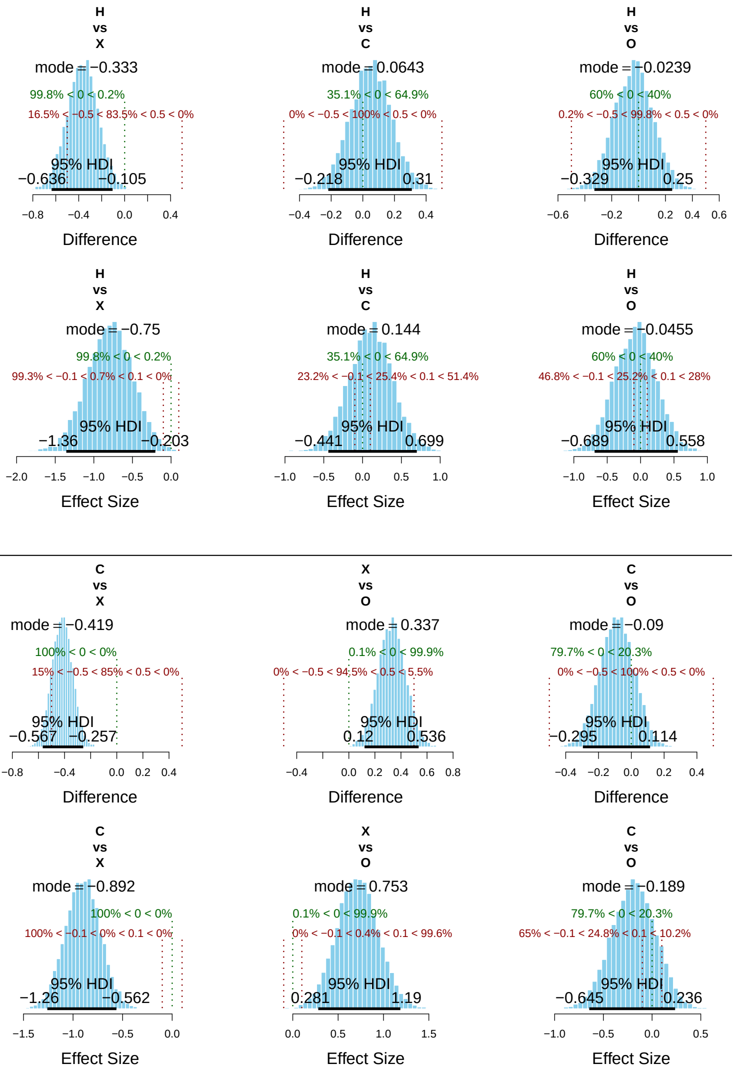




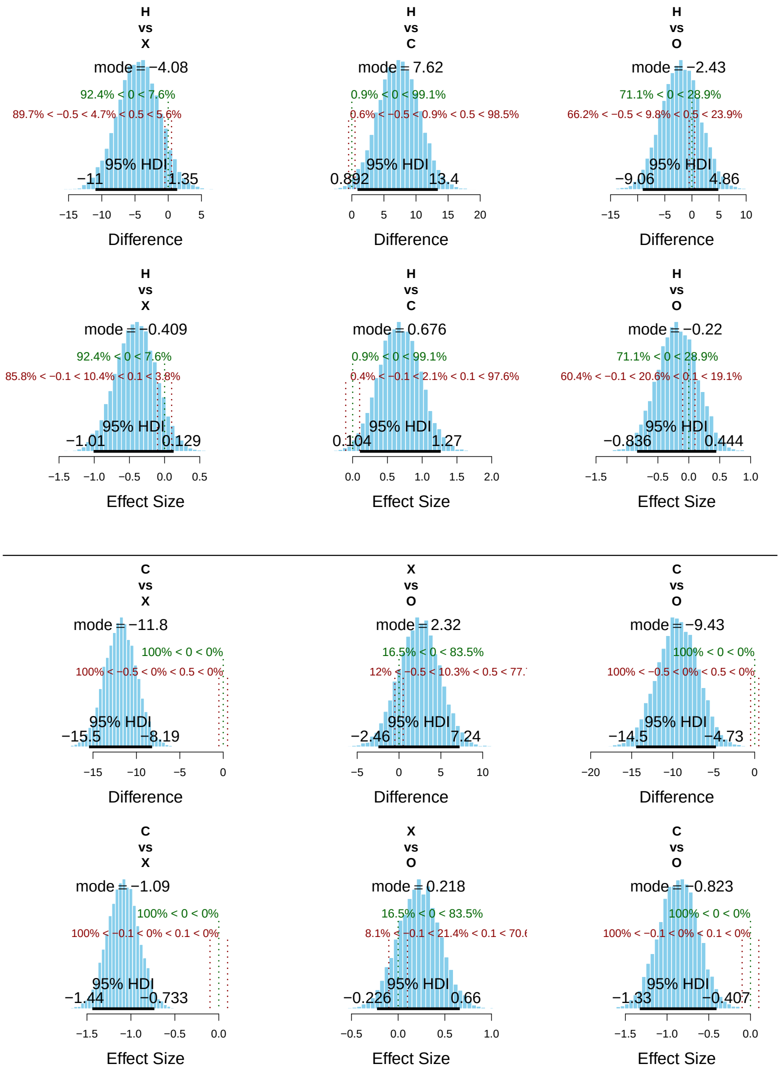
Distribution of differences between groups for metric ILCOM



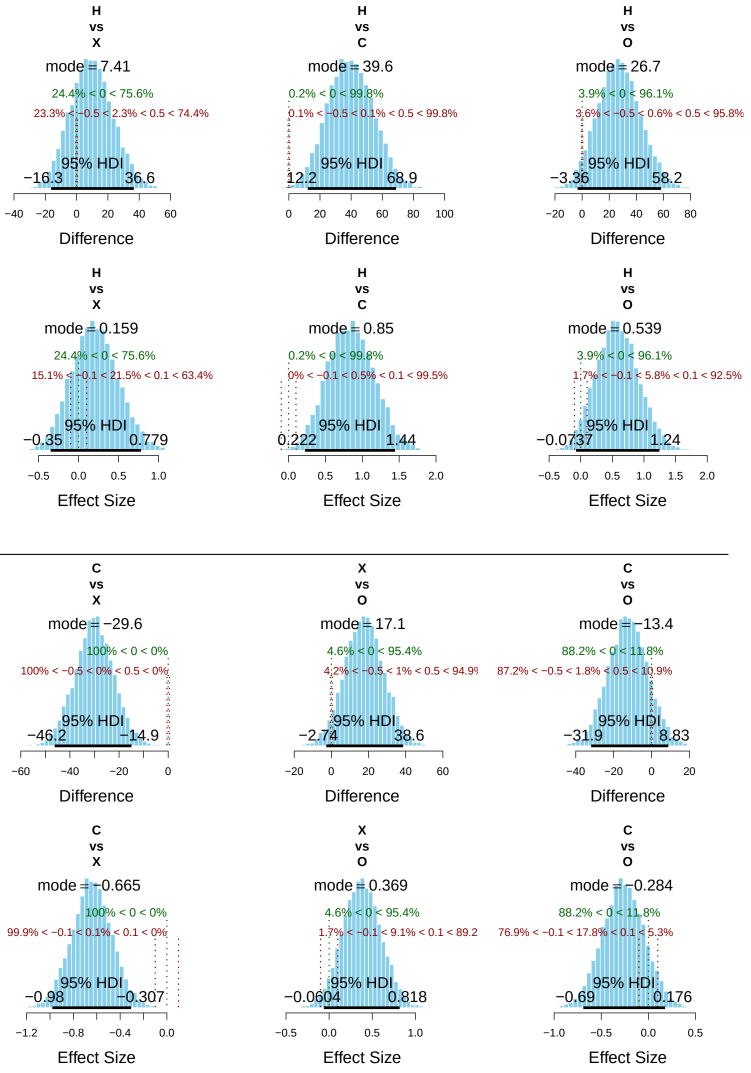
Distribution of differences between groups for metric LCOM



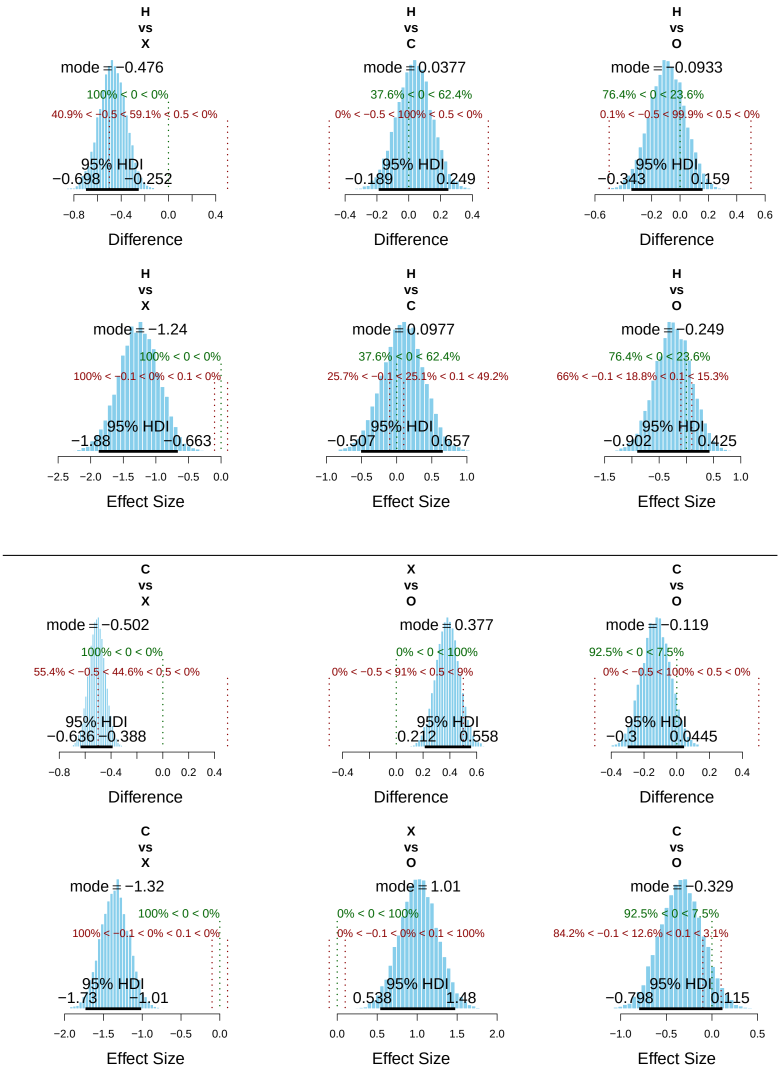
Distribution of differences between groups for metric LD



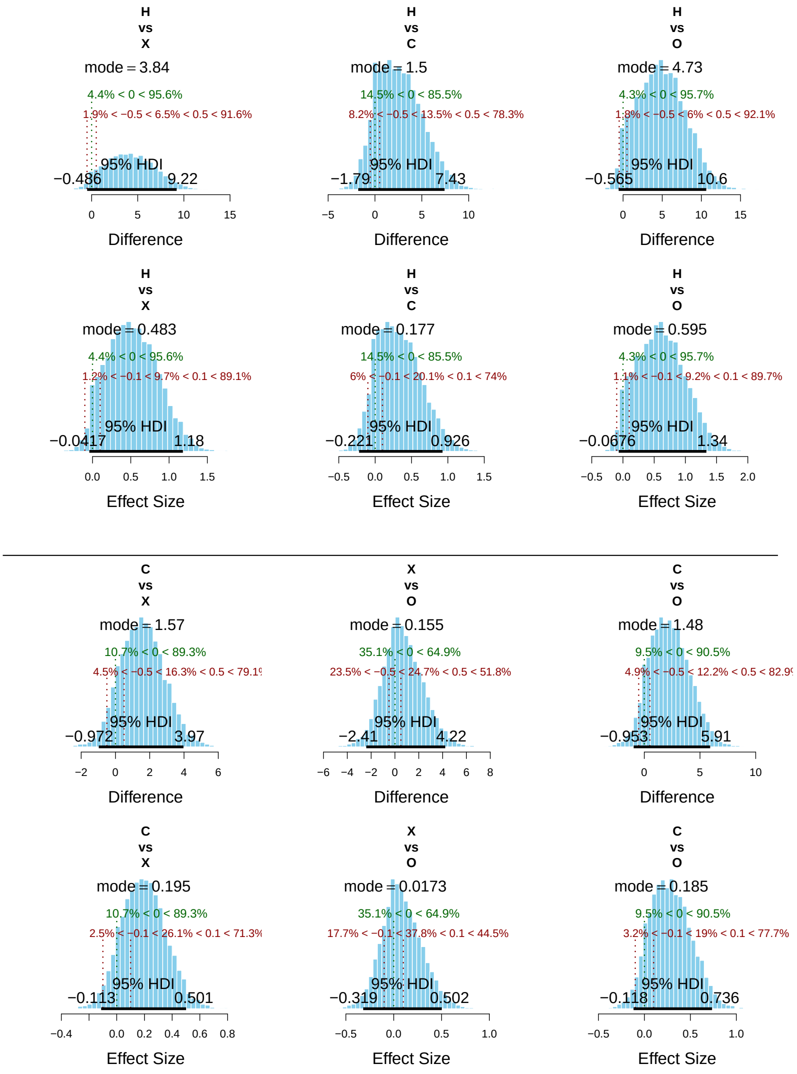
Distribution of differences between groups for metric LEN



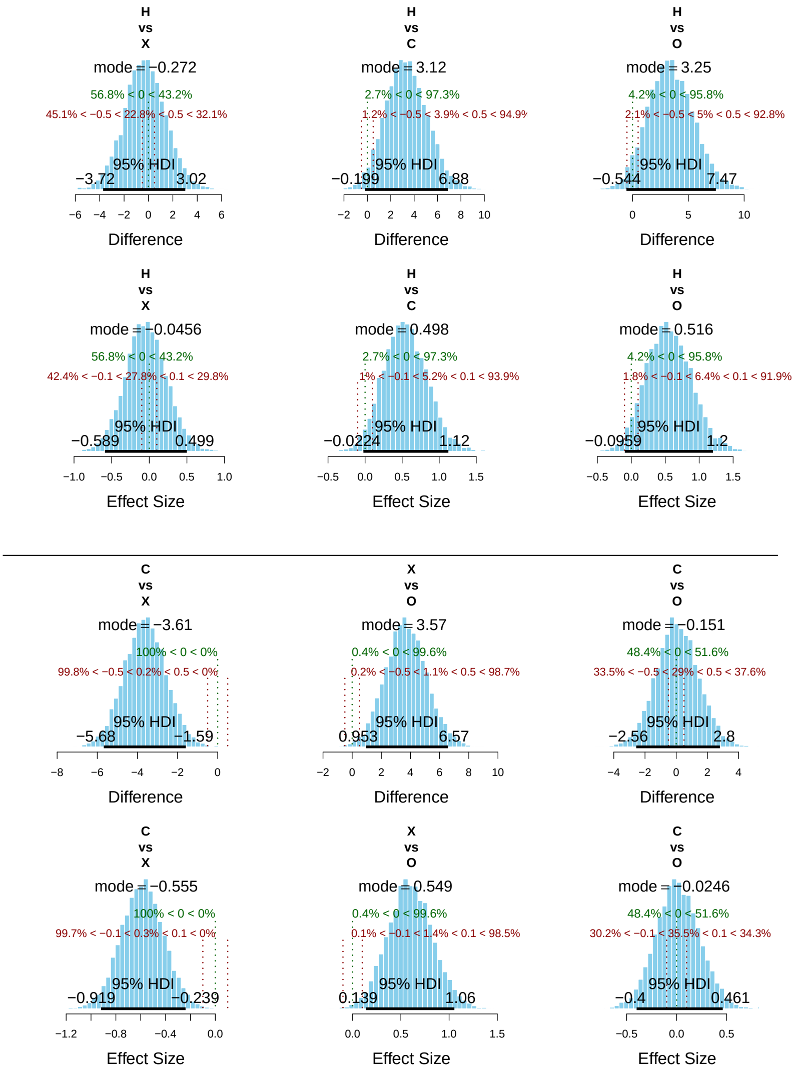
Distribution of differences between groups for metric LOC



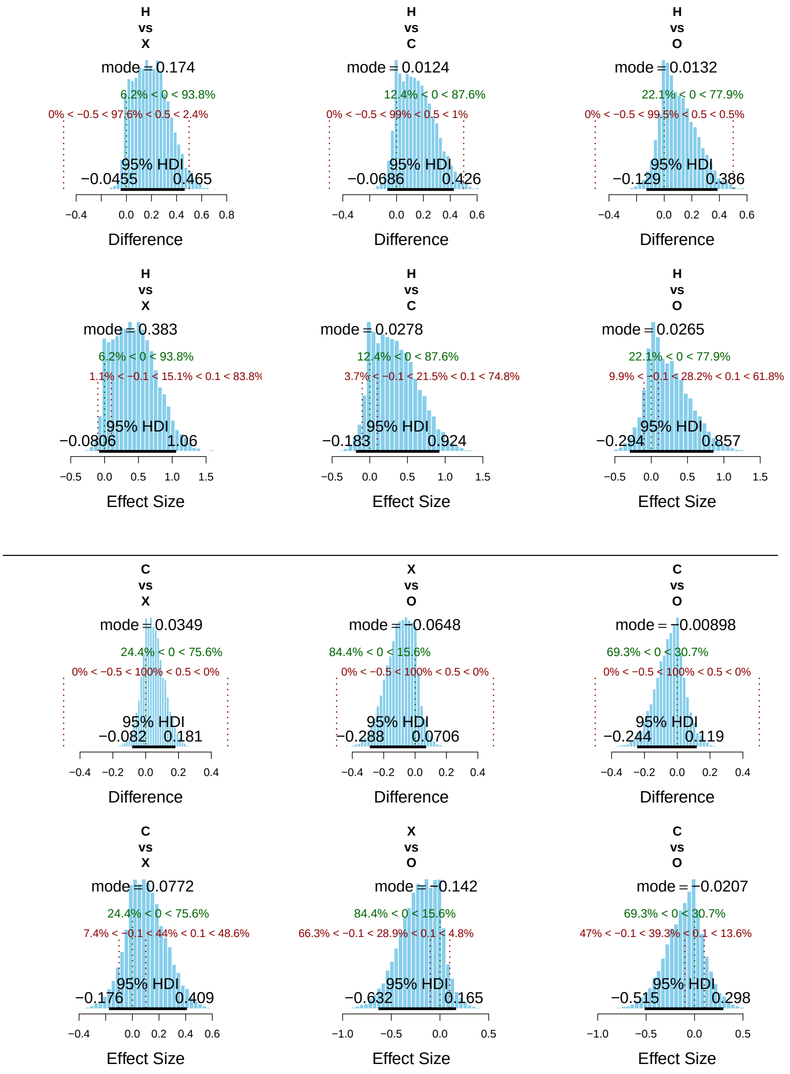
Distribution of differences between groups for metric LOD_Class



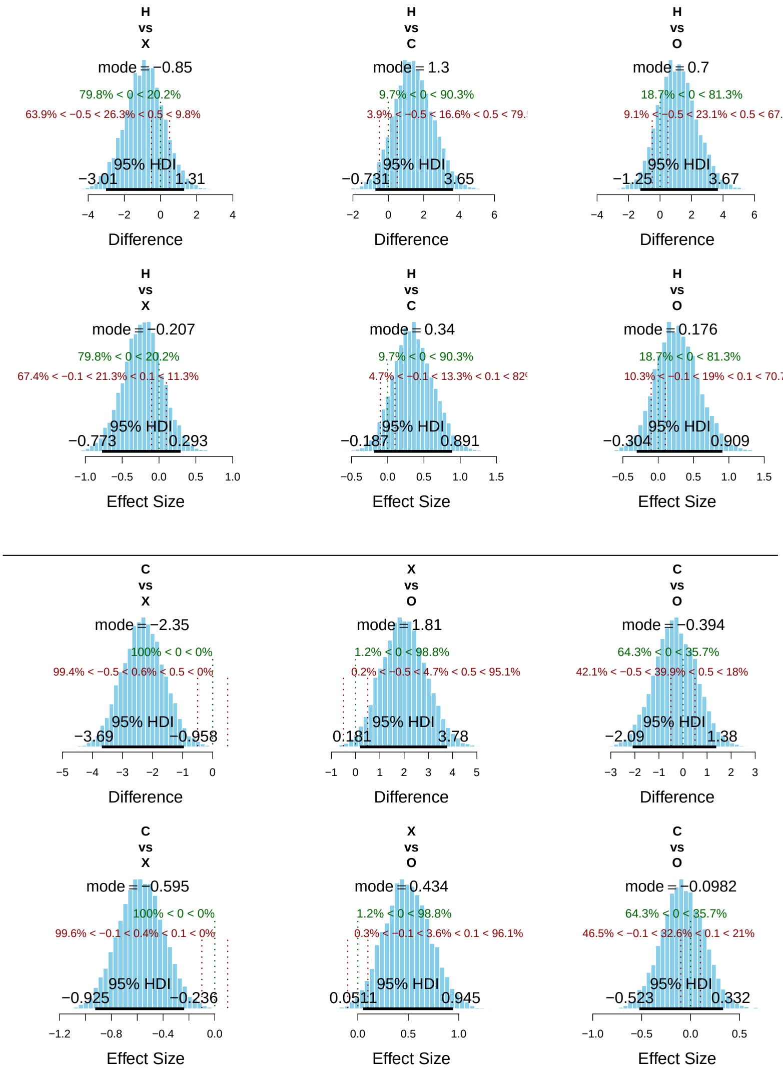
Distribution of differences between groups for metric MPC



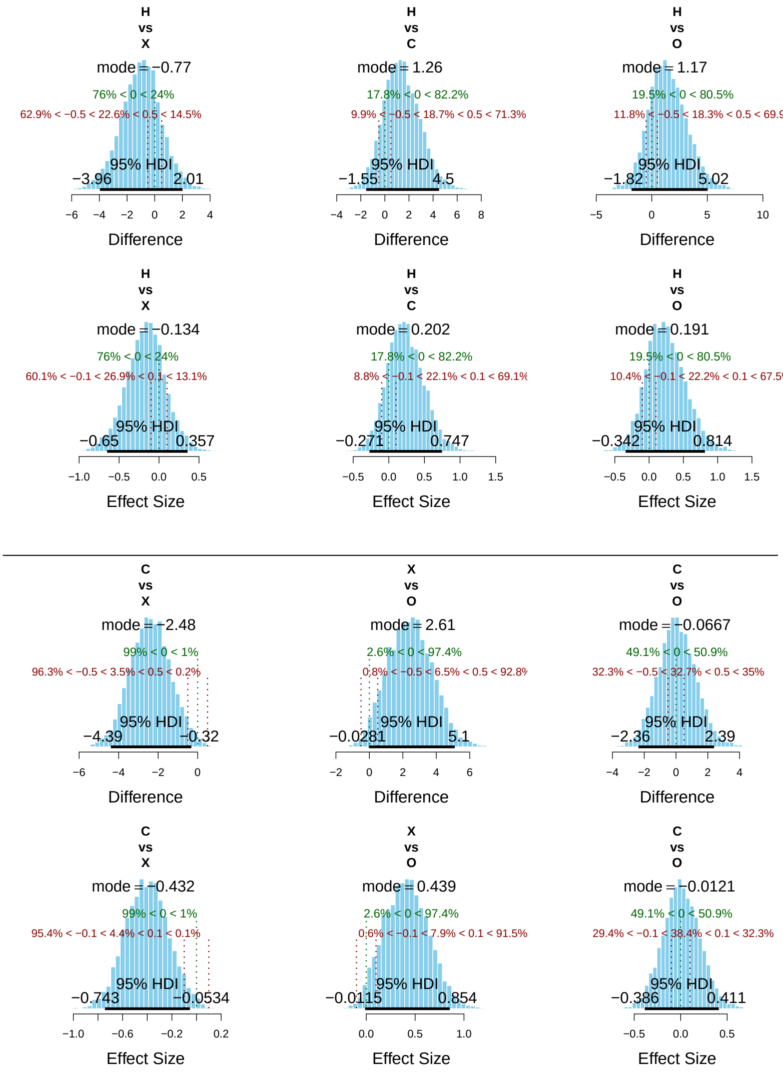
Distribution of differences between groups for metric NAM



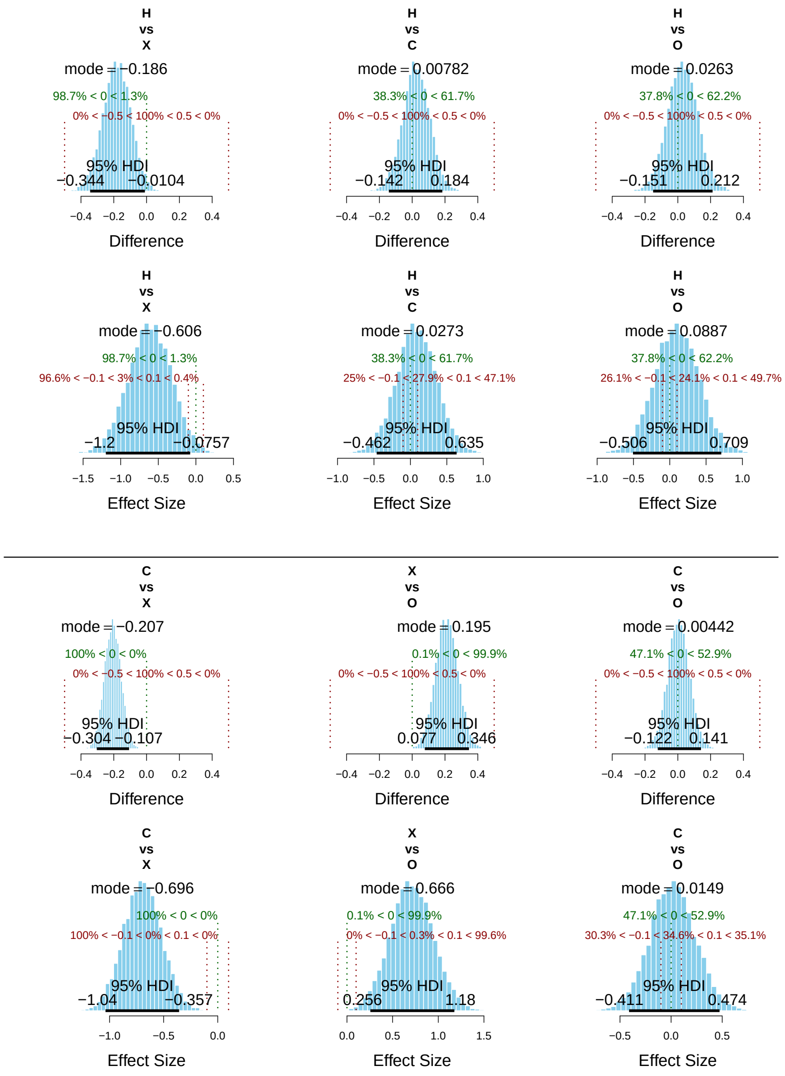
Distribution of differences between groups for metric NOC



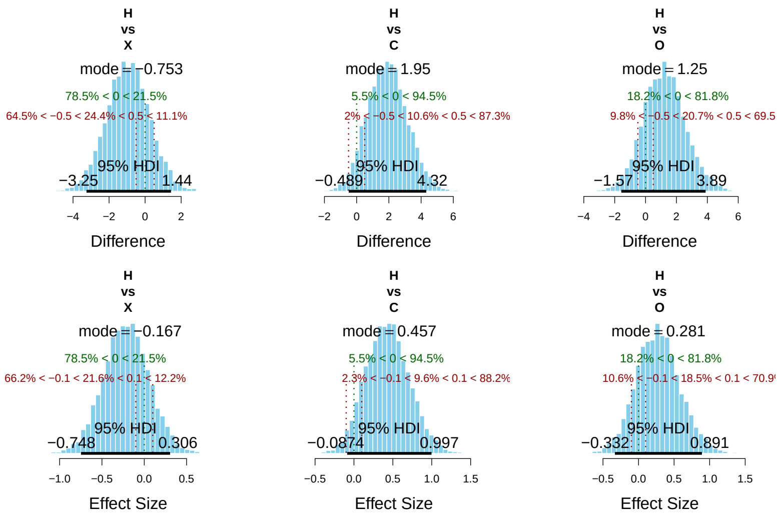
Distribution of differences between groups for metric NOM

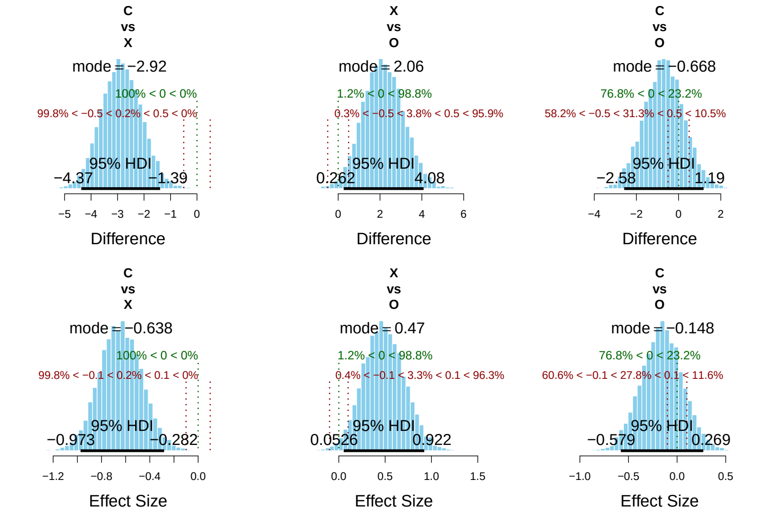


Distribution of differences between groups for metric RFC



Distribution of differences between groups for metric TCC





Distribution of differences between groups for metric WMC; model assumes normal distributions with homogeneous variances posterior predicted curves. These outliers are also outliers of the data. And – apart from the two metrics DIT and NOC – the spreads of the posterior predicted curves seem to reflect the width of the data, too.

4.5. Interpretation and Conclusion

The histrogram (see Fig.4.1) shows that in our dataset not each personality trait is represented. This is a weakness in our investigations, which could not be overcome.

The distributions of the simulated data in many cases seem to be similar (see Fig. 4.3). Only for the metrics LD and LOD_Class there seem to be different distributions between some personality traits when you compare the peaks of the normal curves which suggests that for some metrics there are differences in personality traits. These differences become obvious when we look at the calculated values.

Therefore we have also calculated the differences of the distributions between the groups for each metric. We needed a benchmark to see if these differences are of heavier weight or not. Regarding benchmarks or tresholds for object-oriented metrics, Lanza states: "there is no such thing as a *perfect* treshold." [LM06, p.13]. Thus we were inspired by the procedure of Lanza et al. [LM06] and calculated the arithmetic mean ($\bar{x}$) and standard deviation (σ) of each metric by taking the measured metrics of each measured class as a basis. Results of this calculation can be found in Table 4.3. We then determined the minimum ($min\{\bar{x}, \sigma\}$) of both values and used this value as the basis or benchmark for our evaluation.

As the number of evaluated classes was 179 and the values of the metrics of single classes differed strongly, the standard deviations are relatively high. The mean is always smaller than the standard deviation. Therefore we compared the difference between groups in all cases with the mean.

If the amount of the difference of groups which is represented in the value given by "mode" in Figure 4.4 is higher than the calculated minimum, we define this as an essential difference between groups.

As we can derive from Figure 4.4, there are substantial differences between groups for the metrics DAC, DIT, ILCOM, LD, LEN, LOC, LOD_Class, MPC, NOC, TCC and WMC. These differences are distributed among the different personality traits as shown in Table 4.4.

People with Extraversion as prevalent personality trait differ most often from people with other prevalent personality traits.

Table 4.3.: Calculated arithmetic means and standard deviation of each metric

Metric	$\bar{x}$	σ	$min\{\bar{x}, \sigma\}$
CBO	1,05	2,14	1,05
CYC_Classes	0,66	1,92	0,66
DAC	0,97	1,91	0,97
DIT	0,04	0,27	0,04
ILCOM	0,59	1,23	0,59
LCOM	24,16	84,41	24,16
LD	0,23	0,50	0,23
LEN	7,68	12,08	7,68
LOC	30,82	48,99	30,82
LOD_Class	0,30	0,44	0,30
MPC	2,84	8,08	2,84
NAM	4,14	6,41	4,14
NOC	0,06	0,45	0,06
NOM	2,61	4,17	2,61
RFC	3,77	6,02	3,77
TCC	0,14	0,31	0,14
WMC	2,98	4,60	2,98

The DIT metric is substantial lower for the X group than for the H and O group. As DIT is "highly inversely related" to analyzability, changability and testability and "inversely related" to stability [DP08], people with Extraversion as prevalent personality trait seem to produce code which is better analyzable, changable and testable as well as stable than code which is produced by people with Honesty-Humility and Openness to Experience as prevalent personality trait.

The LD metric is substantial higher for the X group compared to the H, C and O group. As LD is "highly directly related" to analyzability, changeability, stability and testability [DP08], this again underlines the conclusion we drew at metric DIT and in the comparison even includes people with Conscientiousness as prevalent personality trait.

The LEN metric of the X group is substantial higher compared to the C group. As LEN is "highly inversely related" to analyzability, changeability, stability and testability [DP08], with regard to the metric LEN people with Extraversion as prevalent personality trait produce software wich is less analyzable, changable, testable and

Table 4.4.: Substantial differences between groups for each metric

	H–X	H–C	H–O	C–X	X–O	C–O
CBO						
CYC_Classes						
DAC				X	X	
DIT	X	X			X	X
ILCOM		X		X		
LCOM						
LD	X			X	X	
LEN				X		X
LOC		X				
LOD_Class	X			X	X	
MPC	X		X			
NAM						
NOC	X					
NOM						
RFC						
TCC	X			X	X	
WMC				X		

stable compared to people with Conscientiousness as prevalent personality trait. This result is in contrast to the first two results.

The LOD_Class metric of the X group is substantial higher compared to the H, C and O group. As LOD_Class is "highly inversely related" to analyzability, changability and testability and "inversely related" to stability [DP08], people with Extraversion as prevalent personality trait produce code which is less analyzable, changable and testable as well as stable than code which is produced by people with Honesty-Humility, Conscientiousness and Openness to Experience as prevalent personality trait. This underlines the conclusion drawn from the previous metric but contradicts the conclusions drawn from the first two observed metrics.

The MPC metric of the X group is substantial lower compared to the H group. As MPC is "highly inversely related" to analyzability, changeability, stability and testability [DP08], people with Extraversion as prevalent personality trait seem to produce code which is better analyzable, changable, testable and stable than code which is produced by people with Honesty-Humility as prevalent personality trait. This again – in contrast to the previous two findings – supports the findings of the first two metrics, only limited to the comparison with the H group.

The NOC metric of the X group is substantial lower compared to the H group. As NOC is "inversely related" to analyzability, stability and testability and "highly inversely related" to changability [DP08], people with Extraversion as prevalent personality trait seem to produce code which is better analyzable, changable and testable as well as stable than code which is produced by people with Honesty-Humility as prevalent personality trait.

The TCC metric of the X group is substantial higher compared to the H, C and O group. As TCC is "highly directly related" to analyzability, changeability, stability and testability [DP08], people with Extraversion as prevalent personality trait seem to produce code which is better analyzable, changable and testable as well as stable than code which is produced by people with Honesty-Humility, Conscientiousness and Openness to Experience as prevalent personality traits.

The WMC metric of the X group is substantial higher compared to the C group. As WMC is "highly inversely related" to analyzability, changability and testability and "inversely related" to stability [DP08], people with Extraversion as prevalent personality trait seem to produce code which is less analyzable, changable and testable as well as stable than code which is produced by people with Conscientiousness as prevalent personality trait.

Also people with Conscientiousness as prevalent personality trait differ for some metrics from people with Honesty-Humility as prevalent personality trait. The DIT, ILCOM and LOC metric of the C group is substantial lower compared to the H group. As those metrics are "inversely related" and "highly inversely related" to analyzability, changability, testability and stability [DP08], people with Conscientiousness as prevalent personality trait seem to produce code which is better analyzable, changable and testable as well as stable than code which is produced by people with Honesty-Humility as prevalent personality trait.

We see that for the metrics DIT, LD, MPC, NOC and TCC people with Extraversion as prevalent personality trait seem do produce code wich is better analyzable, changable, testable and stable than code produced by people with other prevalent personality traits. But for the metrics LEN, LOD_Class, and WMC the opposite seems to be the case.

And we see that for the metrics DIT, ILCOM and LOC people with Conscientiousness as prevalent personality trait seem to overperform people with Honesty-Humility as prevalent personality trait.

This leads to the conclusion that in our study we cannot make one general statement about the connection between maintainability and personality. But we can state that there are differences for metrics between different personality traits. In our case we defined maintainability as combination of the 17 metrics above. For the majority of them a positive influence is predicted for people with Extraversion as prevalent personality trait, but for some a negative influence is predicted compared to people with the other prevalent personality traits or subgroups of them.

And we conclude that we can predict maintainability factors of software by personality traits as we have managed to build a model using Bayesian data analysis.

4.6. Limitations and Future Work

Due to external circumstances mentioned above, such as a works council, we were unable to attract a large number of participants. Maybe this could be different in other countries with not as strong regulations in this area as in Germany. So the generalizability of our study results is very limited.

Also some participants contributed only one class we could use for our research, other participants contributed more than 70 useful classes to our research. As a result, some personality traits have a stronger weight in the evaluation. But on the other hand we had to use all data with metric differences unequal to zero because this helped us to improve the estimation of distributions of metrics.

Because of the low number of participants we do not have a representative for every personality trait of the HEXACO personality model. Thus we cannot draw any conclusions about the prediction of maintainability by people whose prevalent traits are Emotionality and Agreeableness.

All these issues mentioned above could be overcome by a greater number of participants. A larger number of participants would also enable the use of causal analysis methods.

Our study was not a controlled experiment but had an observational character. Therefore we could not "control" participants in their measurement. We have defined key points for the measurement but we could not verify that they were being adhered to.

Our study incorporated only two companies. This might have led to a bias of personality traits, as companies (e.g. through human factors division) might already be biased about the personality traits when hiring developers.

Next studies should incorporate more participants and more companies. Thus some sources for bias could be reduced and causal analysis could be conducted. Also studies across countries should be conducted to find out if cultural aspects have an influence on the distribution of personality traits among developers.

In our study, we used the individual maximum values of the traits (and therefore one single characteristic trait) for the analyses. Further studies could break down the single results of each personality trait and investigate the correlations between more or less strong characteristics of the individual personality traits and maintainability.

In this chapter we have presented a model that is filled with only a few measuring points in the context of this work. However, this model can be enriched and adjusted with more and more data over the course of time, so that the prediction of the relationship between personality and maintainability becomes more and more precise.

In the course of our work we have included the qualitative aspect of the model from the very beginning and defined or analyzed it so that the qualitative aspect of the observation can be considered part of the model from the very beginning and so that a comparison between quantitative data input and the results of the qualitative investigation is always possible. In the next Chapter we therefore present our qualitative investigation.

Chapter 5

How to Staff Software Engineering Team Roles Using the Concept of Personality?

An Exploratory Study

As mentioned in the previous Chapter, our investigation also includes a qualitative part in order to be able to consider the qualitative aspect of the model from the very beginning and to enable a constant comparison with the quantitative part of the model. In this Chapter we present our qualitative analysis. Software development is mostly teamwork nowadays. There are studies on criteria according to which teams should be put together to be successful, e.g., personality, skills, roles, ability .

Personality was found to be one significantly influencing factor on individual and team performance . Studies have been conducted which describe how the different roles of a software engineering team should be staffed with respect to personality . But these studies either use unreliable and invalid personality tests or the studied subjects are students which makes it difficult to generalize the results to professionals, since personality changes with major life events like the first job .

Our goal is to provide practitioners with scientifically sound guidance on how to optimally staff a software development team with regard to personality. Therefore we must first of all proceed exploratively in order to generate hypotheses. Thus our aim in this Chapter is to propose hypotheses how the different roles in a software development team – derived from the software lifecycle: project leader, requirements engineer, architect/designer, and developer/tester/maintainer – should be staffed with respect to personality traits to form a high quality team.

5.1. Introduction

Software development nowadays in most cases is teamwork [LL13][p. 83]. Thus the quality of the software strongly depends on the appropriate composition of the development team [AGH+15]. But according to which aspects a software development team should be composed?

Many factors have been studied which might influence team performance, not only in the area of software engineering. Researchers investigated the influence of individual capabilities using brain-based measures [WHJ+07], distinct cognitive abilities [WGC+08], mental models [MHG+00], team role preference and cognitive styles [FWW98]. One factor that was found to have a strong influence is personality [BKPB13; WC17].

Borman and Motowidlo propose a model for defining performance. This model consists of "two components at the highest level: task performance and contextual performance" or citizenship performance where "contextual performance consists of activities that support the broader environment in which the technical core must function, including behaviors such as volunteering for tasks not formally part of the job, demonstrating effort, helping and cooperating with others, following organizational rules and procedures, and supporting organizational objectives" [RB03]. "Both, task performance and citizenship performance are taken into consideration when supervisors evaluate others' performance" [RB03, p.90]. Researchers found that contextual performance "is better predicted by personality variables" [RB03].

Personality traits are a common and also effective method for assembling teams [BOJH08; JH97; NWC99; Rut01; SPWK07; TB03]. "Selecting team members with desirable personality traits may thus be helpful for improving individual performance in team settings" [RB03, p.185]. Also in the area of software engineering, many studies have been conducted which investigate the influence of personality on team performance. Gorla and Lam e.g. used the Myers-Briggs-Type Indicator (MBTI) [TB18] and found that the performance of the team is better if the team leader is N(Intuitive) and F(eeling), the system analyst is T(hinking) and the programmer is E(xtroverted) [GL04]. There have been attempts to map distinct personality traits to software development roles. Capretz and Ahmed mapped "soft skills and psychological traits to the main stages of the software life cycle" and present individual personality profiles for each role [CA10a]. For describing personality they also used the MBTI. And Shoaib et al.[LAA09] found Extraversion to be beneficial for exploratory testing. The problem with these studies is that most of them use either personality theories or tests whose reliability or validity are questionable, like the MBTI [Pit93] or the investigated subjects are students. Our study aims at improving both by using a reliable and valid personality type test and incorporating experts from the IT sector.

In the end, our research proposes hypotheses how the different roles in a software development team – project leader, requirements engineer, architect, and developer/tester/maintainer – should be assigned with respect to their personality traits using the HEXACO personality model. This model divides the human psyche into six dimensions – Honesty/Humility, Emotionality, Extraversion, Agreeableness, Conscientiousness, and Openness to Experience [Ash18]. Further empirical studies could use these hypotheses and test them and thereby build a model which displays how strongly individual personality traits affect individual and team performance.

5.1.1. Research Objectives and Contributions

Our exploratory study aims to hypothesize how a software engineering team should be composed in terms of personality types from the point of view of professionals working in the IT industry to be a successful software development team. The contribution of this study are hypotheses for individual personality profiles for the roles of a project leader, requirements engineer, architect, and developer/tester/maintainer. These hypotheses can and should be tested in further empirical studies.

5.2. Background

Many researchers have tried to find relations between personality traits and job performance [AO15]. Wyrich et al. investigated the influence of personality, happiness, experience and academic performance and – among others – found a negative correlation between conscientiousness and coding challenge performance [WGW19]. A review on literature which describes these links can be found in [KW]. Karimi et al. used the Five Factor Model for the synthesis of their data and ”concluded that there is an indication that personality affects programming but this relation is not clear and more studies are needed to clarify the influence of personality on programming" [KW]. With our study we make a contribution to open up this field. Our point of view is even wider and covers the entire software engineering process beyond programming.

Barrick et al. related personality to team effectiveness [BSNM98]. They "used the Personal Characteristics Inventory to asses the Five Factor Model of personality" [BSNM98]. They found that teams with certain "team" personality traits and also single team members with certain personality traits form higher performing teams [BSNM98]. We also assume – and our initial investigations support our assumptions – that if the different software development team roles are staffed with people who have a certain personality trait profile, this increases effectiveness in the sense that higher-quality software is produced. Unlike Barrick et al., we use a six factor model.

Both – the five and six factor models – are well-known, well-established theories with corresponding tests whose reliability has been proven [AL07; CMP92; SG96]. We chose HEXACO because with this theory and the corresponding test an additional trait (honesty/humility) can be recorded separately [AL08], which is not the case with five factor model. ”the HEXACO model showed considerable predictive validity advantages over the FFM" (five factor model) [AL08]. This is another reason why we chose the HEXACO model.

Da Silva et al. investigated which criteria are used in industry to staff software development teams and to look for relationships to (among others) project success [dFS+13]. They found that ”the consistent use team building criteria correlated significantly with project success, and the criteria related to human factors, such as personality and behavior, presented the strongest correlations" [dFS+13], which motivates our study. They also found ”that the type of development method used can moderate (increase or decrease) this influence" [dFS+13]. Our study supports this claim.

Just like Karimi et al. [KW], Ashton claims that "in general, it is the Conscientiousness factor of the Big Five that is the best predictor, *across* occupations, of *overall* job performance" [Ash18][p. 215]. Our findings indicate that high scores on the Conscientiousness domain level are not beneficial for every role in a software development team.

For software development teams there has already been a proposal how the different roles in a team should be assigned – amongst others – with respect to personality [ABA11]. André et al. used the Myers-Briggs-Type-Indicator (MBTI) for assigning roles in a software development team. In our study, we did not use the MBTI because its validity is questionable [CMH04; Gar96; Hog07]. At the same time they propose how a software development team should be staffed using the Belbin Team Inventory [Bel10]. Our study does not yet include this. In a first step, we generally concentrated on the team and the competences required in it and did not yet assign a team theory. But this can and should be done in the next steps.

Other authors analyzed which further factors, such as workplace design, affect job performance [DL14]. We did not take into account these factors but are aware that they also might influence effectiveness. As we interviewed experts from very different enterprises and branches we assume that this factor will be somewhat offset by the great diversity of the interviewees.

Models have been built which should help to staff software development teams ”at the level of software skills" [ZOY14]. Our study shows that software skills are one important factor for optimal staffing but seem not to be the only one. Personality and team factors seem to play a role that is at least as important, if not even more important.

5.2.1. The Concept of Personality

As our study incorporates the concept of personality, the following section gives a brief overview on personality or personality traits which are mentioned in this article.

The definition of "personality trait" is as follows:

"a **personality trait** refers to differences among individuals in a typical tendency to behave, think, or feel in some conceptually related ways, across a variety of relevant situations and across some fairly long period of time." [Ash18][p. 29].

There are many different types of personality theory and related tests. One instrument which was widely used in the early investigations of personality in software engineering is the Myers-Briggs-Type-Indicator (MBTI) [TB18]. The MBTI is a test based on Jung's theory of personality [Jun21]. But its reliability and validity has been questioned [CMH04; Gar96; Hog07], thus we do not delve into this test.

Other personality models which emerged from the beginning of the 60s and spread from the beginning of the 80s are the Big Five theory or the Five Factor Model. These theories roughly summarize the human psyche in five areas – Extraversion, Agreeableness, Conscientiousness, Emotional Stability (vs. Neuroticism), and Intellect or Imagination/Openness to Experience [Ash18]. From the beginning of the 2000s the HEXACO personality type theory emerged. This theory expands the Big Five theory and divides the human psyche into six dimensions – Honesty/Humility, Emotionality, Extraversion, Agreeableness, Conscientiousness, and Openness to Experience. These "scales have increasingly been widely used in personality research" [Ash18][p. 49]. As the Big Five theory, the HEXACO theory also emerged from a lexical study. HEXACO adds another dimension, Honesty/Humility, which could be a significant factor in our study. Each of these scales itself contains facets – unique aspects of the scales. A description of the HEXACO facets e.g. can be found in [LA09b].

Personality traits are stable for a "fairly long period of time", "it can probably be considered as a period of at least a few years" [Ash18][p. 31]. And "it is possible that even the rather stable, long-run tendencies of an individual might change considerably during the course of a lifetime" [Ash18][p. 31]. This shows that personality is a rather long lasting concept and tasks which suit better to people with certain prevalent personality types do not necessarily have to be done by those people. Also other people are suitable for these tasks, they might just have to train longer because the tasks do not come naturally to them. And the measurement of personality has to be repeated from time to time. Deeper insights into the change of personality across lifetime can be found in [Ash18][p. 85ff].

As far as we know there do not exist studies which use a qualitative method approach and investigate how the roles in a software development team should be staffed using the HEXACO personality theory.

5.3. Methods

Our intention was to explore with which personalities the different software engineering roles have to be staffed to obtain a high quality team. We assumed that for a successful software development team the different roles demand different personality traits. Because our study has an explorative character, we chose a qualitative research approach and conducted an interpretive case study [RH08] by leading semi-structured interviews.

5.3.1. Interviews

For the composition of the set of our interview partners we used criterion sampling [Bry16][p. 409] with professionals of the IT industry which had at least 2 years of working experience in the IT industry. We contacted them personally and asked for their participation. All interviews were recorded on a dictaphone and transcribed afterwards.

We constructed a questionnaire, which helped in guiding the interviews. Participants who were not able to answer the questions in an interview, had the possibility to fill out the questionnaire. It consisted of 29 questions, 6 of them were personal questions like age, gender, description of the tasks at work etc. Some questions concerned their entry into professional life, and the two questions which are important for this work were:

> On which points would you judge the ”quality" of a software engineer? Why? Where software engineer is a person who covers the whole software development process, from requirements engineering to design to programming, testing and maintaining.
>
> How much do you think the ”quality" of the team members influences the quality of the software? Please give reasons for your answer.

Both questions were preceded by questions about successful and failed projects to have already initiated the thought process in the desired direction. We were aware that the question of a person’s quality seemed strange and were therefore curious to see how the interviewees perceived these questions. The questionnaire was also scrutinized by social science experts and is available in the Supplemental File.

We conducted semi-structured interviews which means that we prepared a questionnaire which should guide us through the interview, but if any new aspects were mentioned this form of interview is open to incorporate new questions or change questions, depending on the new insights which are gained. Therefore a "classical" validation of the reliability and consistency of the questionnaire is obsolete because the questionnaire may change from one interview to the next [Fli17].

In the time period between May and July 2012 we led 12 interviews. We granted the interview partners full anonymity and participation was voluntary. Our aim was to increase the likelihood that the questions will be answered honestly and freely. One of the interviewees answered the questions in written form by answering the questionnaire which was used as guideline for the other interviews. 9 interviews were led via Skype[1] or telephone and 2 interviews were conducted face-to-face.

The duration of the interviews varied between 30 minutes and more than 3 hours depending on the personality of the interview partner – some were talkative, others ”just" answered in short sentences. We recorded the interviews with a dictaphone. For this purpose, at the beginning of each interview we asked the interview partners if they agree with the recording. All of the interview partners agreed. We explained what we are planning to do with the interviews.

After transcribing the interviews[2], they were coded open and axial according to [CS15].

5.3.2. Interview Partners

The group of interviewed people was composed of 10 male people and 2 female people. The age ranged between 29 and 41 years with 2 to 15 years of experience in the IT industry. The interviewed professionals cover a wide range of IT sectors: finance, automotive, telecommunication, security, business management, municipal facilities, IT service provider, and the event sector. One part of the enterprises operates only in Germany or Europe and another part is acting worldwide. Also the number of employees of the involved enterprises ranged between 50 on the one side and more than 500 on the other side and therefore cover small enterprises as well as large corporations. Our interview partners cover all mentioned software engineering roles. Thus our results should be applicable for a broad range of software developing companies. The interviewees reported of different development process models, the interpretation of our results takes this fact into account.

Table 5.1 gives an overview of information about the interview partner: which branch of software industry, age, gender, size of the enterprise (number of employees), years of working experience in the IT industry as well as a code which links the interview partner to quotes in the following text.

[1]https://www.skype.com/de/

[2]The transcripts of the interviews are in German.

Table 5.1.: Overview of the interview partners

code[1]	branch[2]	age	gen[3]	role[4]	emp[5]	exp[6]
[BC]	finance	30	m	pl	250-500	10
[TA]	business management	29	m	arch + dev	> 500	2
[CT]	municipal facilities	32	f	apm	> 500	5
[AC]	finance	35	m	dev	250-500	10
[AL]	event sector	30	m	dev	> 500	5
[CE]	IT service provider (broad field)	41	m	pl	> 500	15
[FO]	security	28	m	dev + pl	50-250	2
[JA]	automotive	36	m	dev	> 500	2
[NC]	telecommunication	30	m	tes	> 500	4
[JE]	business management	33	m	pl + req + arch	> 500	8
[SA]	finance	36	m	req	250-500	7
[ME]	finance	32	f	dev	250-500	10

1: code for the interview partner; *2: branch* of software industry; *3: gen* = gender; *4: role* in the software development process: apm = assistant to the project management, pl = project leader/project manager, req = requirements engineer, arch = architect/designer, dev = developer, tes = tester/maintainer; *5: emp* = number of employees; *6: exp* = years of working experience in IT industry.

5.3.3. Coding

The analysis followed the Grounded Theory approach [CS15; GS10]. Interviews were coded open and axial [CS15].

In sum, we interviewed 12 people. Theoretical saturation [Bry16] was already gained after 11 interviews, nevertheless we lead the 12th interview, because we had already asked the person beforehand and the person agreed in participating. Given the fact that it is difficult to attract good interview partners, we did not want to appear rude. Due to the fact that we already obtained theoretical saturation, we did not ask other people than the 12 we have asked before to take part in our interviews. In addition, Figure 5.2 shows that we also gained data saturation. This Figure presents the results of a similarity analysis which we conducted in MaxQDA 18 Analytics Pro [1]. The chosen similarity analysis checks, whether "the selected codes appear in the document or not" [Ver18]. The darker the green, the more similar are the interviews with respect to the codes. The numbers near 1 also indicate a strong resemblance between the documents with respect to the codes [Ver18].

After coding 10 documents the codings of all documents compared to the total

[1]https://www.maxqda.de/produkte/maxqda-analytics-pro

Table 5.2.: Results of the similarity analysis. Abbreviations in the "Document" row and column are taken from Table 5.1 and indicate the respective interview partner.

Document	TA	SA	NC	ME	JE	JA	FO	CE	CT	BC	AC	AL
TA	1	0,8	0,76	0,76	0,78	0,75	0,71	0,65	0,77	0,71	0,73	0,77
SA	0,8	1	0,88	0,88	0,9	0,86	0,81	0,76	0,89	0,82	0,83	0,85
NC	0,76	0,88	1	0,83	0,86	0,81	0,77	0,72	0,85	0,78	0,8	0,81
ME	0,76	0,88	0,83	1	0,88	0,82	0,78	0,72	0,88	0,8	0,81	0,83
JE	0,78	0,9	0,86	0,88	1	0,84	0,81	0,78	0,9	0,83	0,86	0,86
JA	0,75	0,86	0,81	0,82	0,84	1	0,77	0,76	0,83	0,79	0,84	0,84
FO	0,71	0,81	0,77	0,78	0,81	0,77	1	0,68	0,79	0,74	0,74	0,75
CE	0,65	0,76	0,72	0,72	0,78	0,76	0,68	1	0,76	0,69	0,75	0,73
CT	0,77	0,89	0,85	0,88	0,9	0,83	0,79	0,76	1	0,82	0,85	0,85
BC	0,71	0,82	0,78	0,8	0,83	0,79	0,74	0,69	0,82	1	0,79	0,8
AC	0,73	0,83	0,8	0,81	0,86	0,84	0,74	0,75	0,85	0,79	1	0,84
AL	0,77	0,85	0,81	0,83	0,86	0,84	0,75	0,73	0,85	0,8	0,84	1

number of codings was 99,1%. After coding the first three documents this coverage was 89,9%.

The resulting codes served as the basis for determining the desired personality types of the different roles in a software development team. Therefore the codes were linked to the description of the individual personality types [LA09b]. Coding and leading interviews was an iterative process. We constantly checked whether theoretical saturation was reached [Bry16]. If no, further interviews were conducted.

For triangulation purposes [RH08] two additional researchers coded more than 10% of the interviews. These 10% were drawn by lot. This observer triangulation should "increase the precision of the research" [RH08].

Figure 5.1 shows an activity diagram of the research process.

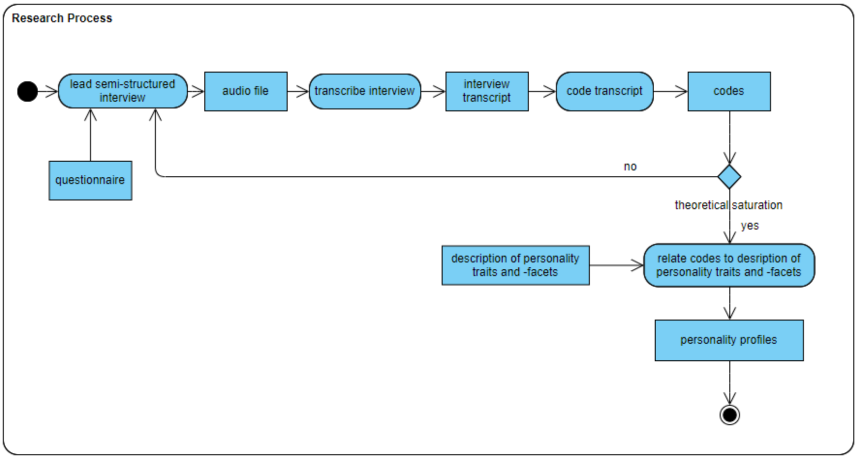


Figure 5.1.: Activity diagram of the research process.

5.4. Results

Individual Team Roles

As the questions did not explicitely mention one special role in the software development process but one general software engineer, the interviewees had the possibility to bring in their experience as much as possible. We interpreted from their description which role they were speaking of and coded it that way. Some interviewees explicitely picked one role and only described desired skills for this particular role.

For open coding the tool MaxQDA version 10 [1] was used. For axial coding, we exported the generated codes of open coding from MaxQDA and imported them in Mindjet MindManager 2016 [2]. This tool supports generating mindmaps and thus for our purposes organizing codes and portray relations. The so evolved codes formed the basis for the mapping to certain personality scales of the HEXACO personality factors. At this stage a change of the language took place. We generated the codes in German and used the English version of the scale description for the mapping. We used the facet-level scale description [LA09b] because the finer structure enabled us a more exact mapping. As mentioned in the Background section, facets are unique

[1]https://www.maxqda.de/
[2]https://www.mindjet.com/de/

aspects of personality scales. Each scale can be composed by several different facets. For the mapping we also used the adjective lists given in [ALG04] and [ALP+04].

The axial coding revealed that the mentioned attributes can also be clustered in competencies. We noticed that when we built clusters with attributes of similar contexts. We then found these contexts to match competence categories. We structured the competencies following [SB04] and [JBFe12].

We were curious to see how the interviewees perceived the question of a person's quality. There was no question on this point. On the opposite, we had the impression, that the interview partners immediately knew what we meant. The questions of the interviewees rather concerned the role or tasks of the software engineer. After talking about that, some interviewees went through the software development process in their head and some picked one role of the software development process and described which qualities the person or people in the respective role(s) should have. No one distinguished between developer, tester and maintainer. Thus we aggregated these roles.

We used the facet level scales of the HEXACO personality inventory and their descriptions [LA09b] to match the adjectives or attributes we found through the analysis of the interviews to personality traits. With this lexical approach we took a similar way as the researchers who constructed e.g. the HEXACO personality inventory [Ash18][p.67ff].

The following Figures 5.2 – 5.5 show our results – hypotheses for the desired personality traits for each role of a software development team. ++ indicates high scores on this facet, + higher scores, - lower scores and - - low scores on the facet.

Here we portray the results from our coding and mapping activities. You will find a discussion of these results in Section DiscussionDiscussion.

5.4.1. Team

The second question which we were interested in was the development team as a whole.

The majority of the interviewees agreed that individual weak members in a team can be compensated by the rest of the team. Weak team members should not be the majority and the team must nevertheless work well together [CE, AL, AC, CT, JA].

> "You can balance it out, so you can have weaker team members with you without the whole product suffering." [AL].

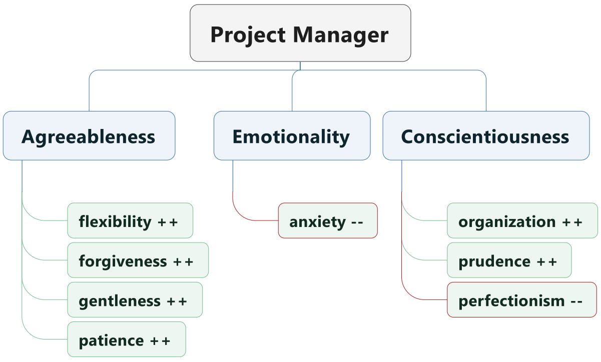


Figure 5.2.: Desired HEXACO personality profile of a project manager.

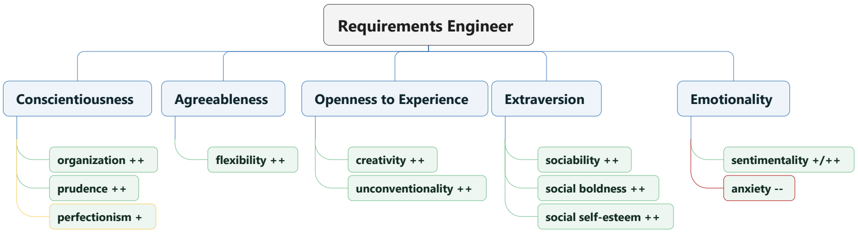


Figure 5.3.: Desired HEXACO personality profile of a requirements engineer.

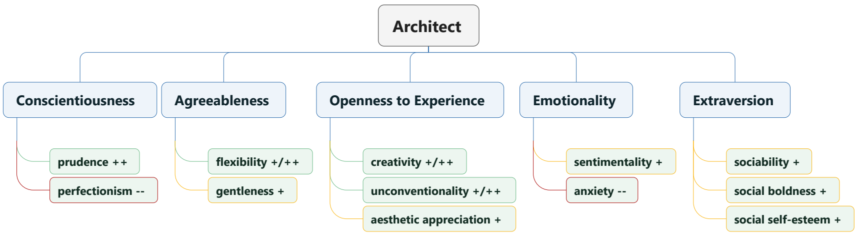


Figure 5.4.: Desired HEXACO personality profile of an architect.

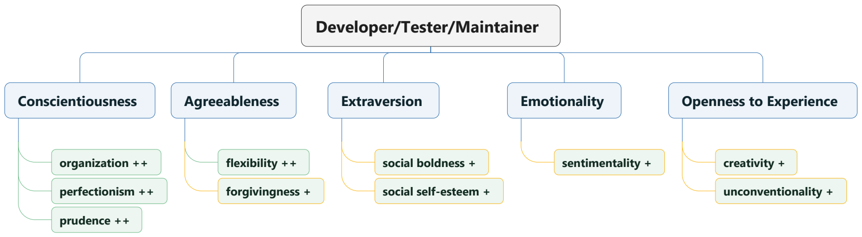


Figure 5.5.: Desired HEXACO personality profile of a developer/tester/maintainer.

> "the team helps at this point a bit, I say, the individual gaps, which one has in its knowledge, which everyone has, that is somewhere obviously, not everyone can know everything, a team can catch, because another perhaps knows well." [AC].

One interviewee on the contrary said that single weak team member can cause malicious software [JE].

> "We've already had colleagues who weren't so technically fit, and that had the effect of not thinking about everything. This means that only this individual case is covered and then leads to the problem as soon as a customer does not run through the intended scenario. Or software is actually built in a way that is very error-prone, because not all outputs are thought of that are possible or are programmed in a way that is very difficult to maintain later, because it is not always the ideal way, but very complicated and complex to program." [JE]

This interviewee uses the waterfall model for software development, the aforementioned interviewees use agile methods. Thus we assume a correlation between the software development model and the extent of ”damage" single weak developers can cause. Interviewees who apply some form of the waterfall model mentioned that individuals can cause great damage whereas interviewees who apply agile development methods all responded that single weak people can be compensated by the team. But one interviewee remarked that agile development methods are not necessarily applicable for all developers [AC]. Some developers do not like having to work so closely in a team.

In the context of talking about teamwork, the interviewee said, "I think there are damn good software engineers who can't do that. But they are then not suitable to develop software with Scrum e.g. It's such a thing, you have to find something else for them." [AC].

The interviews revealed that the chosen process model highlights the strengths or weaknesses of the team members.

> "If a team consists of only good ”players" they do not need that much framework conditions and the project methodology is nearly irrelevant, they will nevertheless produce very good results" [CE].

And a team must have some experts if the product should be good [AC].

> "But if a team is just a bunch of C developers, then they won't write a good Java program, it just doesn't happen without appropriate training etc. and experience. Well, I don't think so. I think that only works as long as you have one or two in there who are really good at what is required of them." [AC].

Talking about the number of weak members in a team,

> "I mean, somewhere, of course, a limit is clear, even from the team size. I think if you have two thirds weaker people with you at the end of the day, and maybe one of them will also take over the management, then this will certainly have a massive negative impact on the software quality. But this is not automatically the case. So I think this can be well balanced by an appropriate project management." [AL].

Thus individual weak team members can be compensated but the majority of the team should be fit in their field of expertise.

There has to be a team spirit. All team members should have the wish to achieve the goal together [ME], [JE].

> "Only people who do a good job and want to achieve a common goal can develop a uniform, well-functioning software." [ME].
>
> "Clearly, one person alone will hardly be able to develop large software anywhere these days, so especially with us you can always see that each task is always distributed among several colleagues. Only if you then work together successfully and consciously in a team you will be successful." [JE].

Thus the capability for teamwork is essential. Team members should get on well together. This does not demand an active part from them, which makes it easier for those who do not like working in a team as mentioned above. The team will be strong if the members are able to ”play" together [CE]. This shows that certain demands are made on both the individual team members and the team leader.

Conscious cooperation is required of every team member [JE]. This means that each team member has to be willing to compromise [FO]. They have to be open for the professional needs of the other team members and be open to each other’s interests [BC]. This includes comprehensive thinking, not only in one’s own area [BC].

They should be reliable, prudent and active thinking [AC, BC].

> "to be able to imagine where there could be difficulties in any form" [SA].
>
> "where there could be problems, not only in their specific area but also perhaps across the board" [BC].
>
> "I must always be aware that what I do has possible effects and I must have considered these effects beforehand. I must have talked to someone from next door at least once, if I touch anything that changes my results or something. I have to know that this is right for the others, too, and that’s something that either goes a bit in the direction of customer orientation or, yes, I know that’s difficult to verbalize, but I think that’s just what you need, a prudence, I think that’s what you need." [AC].

Exaggerated self-confidence and a one-man-show are very hindering [FO]. Team members must be able to communicate in the sense that they can express and accept constructive criticism, have the ability to address problems, listen and communicate [FO, BC]. Every single team member should be "willing[...] to produce something good" and have the claim to do a good job [JA, ME].

Concerning expertise, the interviewees required different levels. Some said that every single team member should know the *basics* of her/his field [JA]. Others required *very good expertise* in her/his field [FO]. The team leader has a key function and must not be weak [SA, NC].

> "The team can only work as well as the team leader works, and if the team leader doesn’t work, the whole team won’t work." [NC].

The team is only as good as the team leader if she/he is the decision-maker [AL, NC].

> "Even a very good team that works well together can certainly compensate a bad project manager. Um, in a way. But I do think he usually plays an important role. Especially if he just made the decision: which way? What is important? What is being implemented? If he then decides to ignore the requirements, then the end product is certainly bad. So there really is the question: how much initiative do the others take, knowing: ok, he says something to me, but I prefer to look again myself. Or well, if I'm told that, I'll implement it. Then, of course, it goes right through." [AL].

There must be a clear division of roles and clear contact points if questions arise [CT].

> "It just has to be clear how the distribution of roles is, who does what, and to whom one can go, if one is not able to do a thing or where to escalate." [CT].

To be successful in a team, the strengths of the individual members must be recognized and placed correctly. The right combination is important [JA, CE, AC, AL, FO, BC] – with respect to personality and with respect to expertise. There are different skills [JA, FO, CE] and the necessary ones have to be put together in a team [JA, BC, AC].

> "if I only let alpha-animals go at each other, nothing comes out at all. ... exactly what I said before, so I say I need strong people with different characteristics, I need the visionary, I need the team player, I need the complete finisher, I need men, I need women, also one topic, only men doesn't work, only women doesn't work either. " [CE].
>
> "I think it's important to simply have the right and also the different skills in a team" [JA].

Having individual experts in a team is important for success [AC]. At the same time, care must be taken not to create head monopolies [AC].

> "In fact, not everyone in a team has to do everything perfectly. I.e. it may quite be that in the team, and I believe, that happens also completely automatically, that you train specialists in the team. That is also

> completely ok, as long as the team can handle it and is concerned that one does not build up too big head monopolies, it works." [AL].

Every team needs one perfectionist and one pragmatic who meet in the middle of the way [FO, CE]. Every team member should have someone to talk to about ideas [FO].

> "you also need people who can be used to discuss ideas, who may not have the ideas themselves, but who immediately understand what you mean when you discuss new ideas about architecture or how to use the programming language with them. This is also very important" [FO].
>
> "And you need a pragmatist in every team who ultimately wants to avoid abstraction, because that is not pragmatic at first" [FO].
>
> "and then in the end they meet somewhere in the middle, where you have a good balance." [FO].

There should be at least one person in the team who puts her/his focus on quality [AC]. Ideally, there is quality awareness in the team [JE]. At least one person in the team should push the topic of quality [JA]. The understanding of quality should be instilled into each team member [JE]. And the developers should adhere to standards [SA]. Each team needs at least one ”complete finisher" ”who is only satisfied when the last point and the last comma are right" [CE]. But not too many of them otherwise the project will drag on [CE]. In addition, the team mates have to fit together [FO]. One factor one interviewee mentioned was time – which is an economical aspect. Good people understand quickly. Some understand more slowly, repetition takes time (and therefore money) [TA].

5.5. Discussion

As we see in Figures 5.2 - 5.5 we created hypotheses that the roles in a software development process should be filled by different HEXACO personality profiles. Ohter than expected, we did not find adjectives which matched to the Honesty/Humility domain. In the following we describe noticeable weightings, which represent a selection of the results.

Please keep in mind that the demanded skills or traits describe the *perfect* staffing of a software development team which in real life can only be approximated.

5.5.1. Project Leader

As we can see in Figure 5.2, the project leader should have high scores on the Agreeableness domain. The interviewees mentioned qualities which could be mapped to all four facet scales of the Agreeableness domain. The social component as well as methodological skills occupied the interviewees more than professional skills. Interviewees demanded error friendliness, a climate where errors can happen and it is possible to admit them and talk about them without being afraid that 'life will be made a living hell' [FO]. Such a climate helps to find solutions together and to avoid similar mistakes in the future. The opposite would be to remain silent and risk the project to fail. Interviewees also demanded trust in team members.

The Conscientousness domain is apportioned very differently. On the one hand, high scores on the organization and prudence scale are demanded, they should be helpful in leading the project in a forward-thinking and structured way. On the other hand, low scores on the perfectionism scale are demanded which should also help to successfully complete the project. Software which has to be produced in time and budget can never be perfect, thus the project leader has to make decisions that consciously accept mistakes. Here also low scores on the anxiety scale are helpful.

Many of the interviewees mentioned that a team is only as good as its team leader or project leader, if the team has to follow her/his instructions. This also implies expert knowledge, and maybe that is the reason why only one interviewee explicitely mentioned this [CT]. This interviewee demanded expert knowledge because if in meetings the members find out that the project leader does not own expert knowledge, she/he will lose her/his authority. That very authority was demanded also by other interviewees.

5.5.2. Requirements Engineer

As a requirements engineer has to communicate in very different directions, high levels on the Extraversion domain are demanded [BC, CT, JE, ME, NC, SA] (see Figure 5.3). The requirements engineer has to be open to the customer as well as to the development team and perceive the needs of both sides. She/he has to communicate with both sides and thus speak the language of both sides which also implies expert knowledge – on the one hand of the domain of the customer and on the other hand of the questions or problems which developers might have [AC, AL, BC, CT, CE, JE, ME, NC]. The requirements engineer has to be a socializer and keep contact to customers and the development team. She/he has to be able to verbalize her/his thoughts and opinions and to take the opinion.

For finding best solutions for the customer (and the programmer) the requirements engineer might have to consider unusual solutions or think outside the box. Thus high scores on the creativity and unconventionality facet scales are demanded.

On the Conscientiousness domain, high levels on the organization and prudence facet scale are demanded. Circumspection and forward-thinking are requested, because the requirements engineer is the ”place" where serious decisions are made for the developers [AC]. If the requirements engineer misunderstands something the customer said or only meant and maybe did not express, this might have a negative impact on the following development process and lead to increases of time or money which has to be spent for the project. Here also higher scores on the perfectionism scale are helpful.

5.5.3. Architect

For the architect role the destilled personality facets are broad in scope but not deep (see Figure 5.4). Higher scores in the Extraversion domain level are beneficial, but compared to the requirements engineer they do not have to be as pronounced. Low scores on the anxiety facet scale and high scores on the prudence facet scale were demanded because if difficulties occur, the architect has to stay calm and rationally think of possible solutions. Beneficial are low scores on the perfectionism facet scale. Just as the project leader, the architect has to make decisions which might accept imperfect software if the project should be finished in time and in budget. One interviewee said, this role is the most demanding role in the software development process because the architect combines the skills of every other role in the software development process, not in the same depth but in any case in width [CE].

5.5.4. Developer/Tester/Maintainer

As we see in Figure 5.5, great emphasis is placed on the Conscientiousness domain level for this role. High scores on the organization, perfectionism and prudence facet scale were demanded by the interviewees. Kanij et al. found that ”software testers are significantly higher on the conscientiousness factor than other software development practitioners” [KMG15]. Our results support this partly, as Kanij et al. state this only for software testers whereas software developers have different specifications of personality traits.

Concerning social skills we found a clash of demands. For developers, the sociability scale is a certain contradiction. Some interview partners mentioned, communication skills are of importance [FO, CT, BC] which would mean higher scores on the sociability scale. Other interview partners said, communication skills are not important, which means that low levels on the sociability scale are not disadvantageous.

A developer should not be a "babbler" [JA]. The majority of the interviewees demanded that the developer should be able to express her/his needs, to address problems openly, and to be open to criticism – as giver as well as receiver.

One interviewee mentioned that a developer should have social manners which enable successful teamwork. Also the opinions on the ability for teamwork varied. Some said this is essential [FO, CE], others do not think it's so important [BC]. We had the impression these demands strongly depended on the software development method – whether the team uses agile methods or some kind of a waterfall model. In any case, a developer should not feel the urge to deliver a one-man show [FO]. These demands we tried to mirror in the personality profile in Figure 5.5. And – unlike the other roles – there was a demand for self motivation. Unfortunately, this skill cannot be matched to a personality facet. We identified creativity and unconventionality as desired facets which supports the research results of e.g. Amin et al. [ARBH15] and Graziotin [Dan13].

Many of the interviewees demanded from developers to have a feeling for the needs of the customer because in the end, the customer – and not the developer – has to like the product. Many interviewees mentioned that the developer should make sure that the interface is appealing to the customer.

5.5.5. Team

From the interviews we can hypothesize that the quality of the software is strongly dependent on the process model during development (Waterfall → single ones can break a lot, agile methods → individual weaknesses are caught). In the first case, you can't afford a weak individuals, but in the second case, you can. Realistically, however, there are weak people, there will very rarely be a team where all team members are strong at everything. Such teams are usually only used for crisis management. Otherwise this would be very expensive from a business point of view. So process models for development should be chosen in such a way that the team can catch the mistakes of individuals if necessary. However, agile process models are not suitable for every developer. This must also be taken into account.

Customer orientation was often demanded, but it was also mentioned that a pure developer might be overwhelmed because she/he often concentrates on the algorithms and functions and not on usability. So ideally there is still one other person responsible for usability. Surface and usability are important [AC, AL, JA].

Understanding of the needs and activities of other team members was often mentioned. This means team cohesion is enormously important, if good software is to develop.

The Gallup Q12 confirms our finding that the team can only be as good as the project leader [Cla17]. Gallup sais that the management is crucial for motivation and engagement in the team. Thus the Gallup questionnaire measures the satisfaction of the team members with the management. Only in Germany the sales could have been increased by105 billion Euros a year if enterprises would have capable executives [Cla17].

5.5.6. Limitations

Our study might have cultural limitations. All interview partners were german. Some of them worked in internationally active companies, nevertheless the perspective might be limited. One interview partner mentioned that in some anglophone countries the background of people who are employed in software development projects is completely different from the german style. There humanists are part of a software development team, too. In Germany, such a constellation is very rare.

The mapping of the skills to the roles was in some cases our interpretation. Nevertheless we assume that the context has made the correct allocation possible.

The execution and evaluation of the study was carried out by one single researcher. Each individual step of this research is documented and can be found in the Supplemental file. So the interested reader can check the individual conclusions for her- or himself. In addition, the researcher conducted an interrater reliability to minimize possible interpretation errors.

The interviewees might have oversubscribed stereotypes. We have tried to compensate for this effect by choosing interview partners with different roles in a software development process. In addition, we have chosen interview partners from a wide range of very different companies.

The judgement of "quality" might be subjective. But in every qualitative study where interviews are conducted, subjective impressions are caught – and this is wanted. People report from their own subjective experiences. Interviewing several different people – and not only one single person – helps to receive an overall picture and to compensate this effect. We interviewed as many people as were needed to obtain theoretical saturation. The qualitative approach of this study fits best because we investigate "only" the opinions of the interviewees. A next step could be an empirical study which examines our hypotheses on the basis of statistical evaluations.

We have not asked if the interview partners have worked with the person they describe as optimal staffing. Thus we do not know if the description is derived from a wish of how to staff or experiences of working together with such a person.

A colleague said, "Ok, now you know how an optimal striker is equipped. Now you buy three Christiano Ronaldos and you find that it's not working." Yes, the colleague is right. We have not explicitly examined the cooperation of the single team roles. But some interviewees incorporated this in their thoughts, e.g. when they demanded no babblers, no one-man shows etc. Our study concentrates on the optimal staffing of each role without focusing on the bonds between the roles. Another study could take a deeper look at the group behaviour if every role is staffed according to our suggestion. There are also group theories like Belbin's team roles [Bel10] which could be included in the investigation.

5.6. Conclusions

During the analysis phase it was not always simple to link personality facets to demanded skills or traits. In some cases it was hard because there was a demand for high values in some sub facets of a trait but at the same time low values for the trait. Additionaly, the same demands, e.g. communication skills, were differently mapped to personality scales, depending on the role. Some of the scales had to have a larger expression for one role than for another. It is important to evaluate the facet scale, because people demand different values on the facet scales. The evaluation of the domain scales does not testify enough.

We found that not every demanded skill could be mapped to a personality trait. Interviewees demanded skills which are assigned to mental abilities, e.g. analytical thinking, having an imagination or distinguishing the essential from the insignificant. Regarding mental abilities ”There does appear to be a high level of consistency across the life course in relative levels of mental ability, at least after late childhood"[Ash18][p. 243]. Because of this, it is very interesting to also consider the concept of mental ability. Further work has to be done in the area of group theory. Our study investigated team factors but we did not yet apply a group theory. There are already studies which include this (e.g. [AO15]) but they do not combine it with the HEXACO model. A combination of personality using HEXACO, mental ability, and group theory as basis for the research for investigating the optimal staffing of software development teams is a field of research that we believe should be pursued.

Also gender aspects should be taken into account. Many researchers explored the optimal distribution of male and female team members to increase the team performance (e.g. [AAI12; FN01; GFRV14; GJOT14; HOv13]). Razavian and Lago investigated software architecting teams and found "traits and skills linked to the feminine role in architecting teams. Much of such expertise relates to the skills required to successfully deal with software architecting's human aspects" [RL16]. Gilal et al. found, that "gender should be kept in the consideration when composing teams based on personality types" to ensure high team performance [GJO+16]. Our research also revealed that if the role of a scrum master is occupied by a woman, this might increase team performance [WB15]. The corresponding investigation is therefore presented in the next Chapter.

Chapter 6

Less Distress with a Scrum Mistress?

On the Impact of Females in Agile Software Development Teams

The Literature Review and the qualitative study have shown that gender aspects should also be considered when putting together a successful software engineering team. That is why in the following we present a qualitative study, in which we have examined what effects it has when the role of Scrum Master is held by women.

6.1. Introduction

All of us encountered the topic gender equality in some form or another. For some time now there are discussions about how to attract more women to work in the

Information and Communication Technology (ICT) field. “The software world struggles with Diversity as it is. It’s a problem for our profession, in that we lose access to talent, and it’s a problem for many women who don’t get the chance to develop a satisfying career in programming.” [Fow09]. “The percentage of women software developers in the U.S. [is] less than 25% today” [Jud12]. In Germany, the amount of female IT experts is only 14% [BIT14]. According to a study by the IT job portal Honeypot, Germany’s share of women in the IT industry ranks 20th out of a total of 41 OECD and EU countries evaluated [Hon18]. And the gender pay gap in the IT sector in Germany is 25% [Hon18]. On the other hand, BITKOM reports a need of 41.000 IT experts in 2014 only in Germany [BIT]. So at least from this point of view the IT branch should try to attract more women.

A “team made of members with different backgrounds, perspectives and motivations is critical for organizational knowledge creation to take place” [Jud12; Non07]. Therefore for ICT and in particular software development projects there might be also performance and quality reasons why more women should be involved. How about performance in diverse, mixed gender teams? Some authors claim that “Diversity often leads to enhanced abilities to perform tasks, greater creativity, and better decisions and outcomes” [KWS09]. But the “actual evidence for the input-process-output linkage is not as strong as one might like. [...] The results for age and gender are [...] mixed, with several studies showing neutral results” [MN05].

Catolino et al. found that “women are instrumental to reducing community smells in software development teams” [CPT+19]. Since agile software development methodologies like Scrum strongly emphasize the role of human factors [Sch97], gender is likely to have an even stronger impact on the outcomes of agile software projects. This was also one result of the Literature Review in Chapter 3.

Therefore, in the present Chapter results of an exploratory qualitative case study are presented, carried out in the context of a students’ software project with two mixed-gender agile teams each developing a smart phone app. Due to the specific curricula of the bachelor study programs in which this project was done, the number of female participants in the case study was reasonably high compared to usual Software Engineering or Computer Science study programs. The results of the case study indicate that there is in fact a strong evidence for the positive influence of female members on the performance of agile software development teams.

The rest of this Chapter is organized as follows: In Section 6.2 the related work is discussed in detail and Section 6.3 explains the design of the exploratory case study. The obtained results are discussed in Section 6.4, while Section 6.5 outlines the current limitations of these results. We conclude with a summary of our findings.

6.2. Related Work

The relation of women to studies and work in the Information and Communication Technology (ICT) field in general as well as with software development in particular has been studied by numerous authors. Various aspects like the female perception of the ICT field [HNB04; Jud12] or the gender impact on the behavior of IT professionals in various contexts (i.e. ICT entrepreneurship, Open Source Software (OSS) projects) have been analyzed with partially contradicting findings [MW05; PHM10].

With respect to software development in particular, the team performance of the development team is an important quality factor [GCF98]. However, the impact of gender diversity on team performance in general has also been discussed with contradicting findings so far:

Ivanova-Stenzel and Kübler found that "relative to a single-sex environment, gender diversity increases the gender performance gap with team pay whereas it decreases the gap with team competition. The results show that there can be a tension between the objective to maximize overall performance and to minimize gender inequality." [IK11].

Apesteguia et al. investigated the impact of gender composition on team performance and decision making. They found that teams formed solely by women were "significantly outperformed by all other gender combinations". But they also found that "the best performing group is two men and one woman" [AAI12].

Hoogendoorn et al. observed other results. Teams "with an equal gender mix perform better than male-dominated and female-dominated teams in terms of sales, profits and earnings per share" [HOv13].

Bear and Woolley claim that "in occupations dominated by males, such as teams of engineers, gender diversity has strong, negative effects on team performance, whereas in gender-balanced occupations, gender diversity has significantly positive effects on team performance" [BW11].

The previous results show that the advantages or disadvantages of gender-mixed teams still remain a controversial issue so far.

Concerning success factors of agile development teams, researchers always incorporate people factors and communication [AJ01; MKK06]. Darwish and Rizk name –

among others – “effective communication and feedback with the user or within the team for each task” as one success factor of agile development [RM15]. Cockburn and also underline that “people working together with good communication and interaction can operate at noticeably higher levels than when they use their individual talents. [...] Therefore, agile project teams focus on increasing both individual competencies and collaboration levels.” [AJ01]. Chagas et al. investigates with the help of a systematic literature review which human factors impact agile projects and found communication, collaboration and trust as the most cited human factors. A study they conducted afterwards among 186 companies identified communication as the most important factor [CSSV15]. Lindvall et al. in a survey found that for a successful agile project personnel is needed which possesses good people and communication skills [LBB+02]. “The three most important success factors are culture, people, and communication” [LBB+02]. This shows that scientists and practical experts agree that both communication and the human factor are of fundamental importance in successful agile projects. Therefore a female management style which “is centered on communication and building positive relationships” [Hag98] can be very supportive for successful agile projects.

Since agile software development methodologies like Scrum [Sch97] strongly emphasize the social nature of the software process and the role of human factors [WB07], agile projects are likely to be stronger influenced by the diversity of the team members [DRG10]. “The agile approach reflects the notion that development environments should support communication and information sharing, in addition to heavy testing, short releases, customer satisfaction, and sustainable work-pace for all individuals involved.” [BFHD07]. In turn, the increasing usage of agile methodologies may increase the role and impact of diversity and gender aspects in software projects as well [HD06].

Organizations that use agile approaches should also adapt the changed required personality traits at management level: “a team-focused organization has a different definition management than a traditional hierarchical one. Rather than command and control, a manager becomes a facilitator and impediment remover.” [Jud12]. “Agile companies practice leadership-collaboration rather than command-control management.” [AJ01].

This perfectly fits to a female management style which “is centered on communication and building positive relationships” [Hag98]. Burke’s and Collin’s findings support this, “Females are more likely than males to indicate that they use an interactive style of management called transformational leadership.” [BC01]. Beranek et al. confirmed this management style of women. They examined “the informal role distribution in student software engineering teams”. They found that women “more often assumed group building and maintenance roles than men”. Another thing that they observed was that although female students did not report low technical skills (leading roles are strongly connected with technical skills), no one of them was assigned the team coordinator or technical coordinator role [BZG05]. They conclude to “encourage female executives, because many women show strengths in programming tasks as well as coordination tasks, which is a desirable combination for executive positions.”

In agile development teams, this female management style seems to perfectly match the role and tasks of a Scrum master, who is responsible for a “working Scrum” ensuring good communication and smooth teamwork, which is crucial for the success of the Scrum methodology [PP11]. This is particularly important given that communication is one of the most important success factors for agile projects, as literature has shown. And communication is one of the factors which distinguishes the female from the male management style.

However, despite the fact that in the literature some evidence exists that agile software development is in fact influenced by gender and may particularly benefit from female team members, published results on this effect are scarce.

Therefore, in the present Chapter the evident question is addressed, if gender diversity has an impact on the performance of agile development teams. Or more precisely: Does the female management style of a Scrum mistress have an influence on the performance of the team?

6.3. Study Design

For this purpose an exploratory case study was carried out within a mobile app development course. The duration of this course is one semester. The purpose is to teach the participants not only different app programming technologies (like native Android development with Java or hybrid app development with HTML5 and JavaScript) but also development methodologies like Scrum and User-centered Design relevant for developing successful mobile apps.

Therefore, the participants need to successfully complete a group assignment in the form of an agile software project, developing a mobile application for a semi-realistic

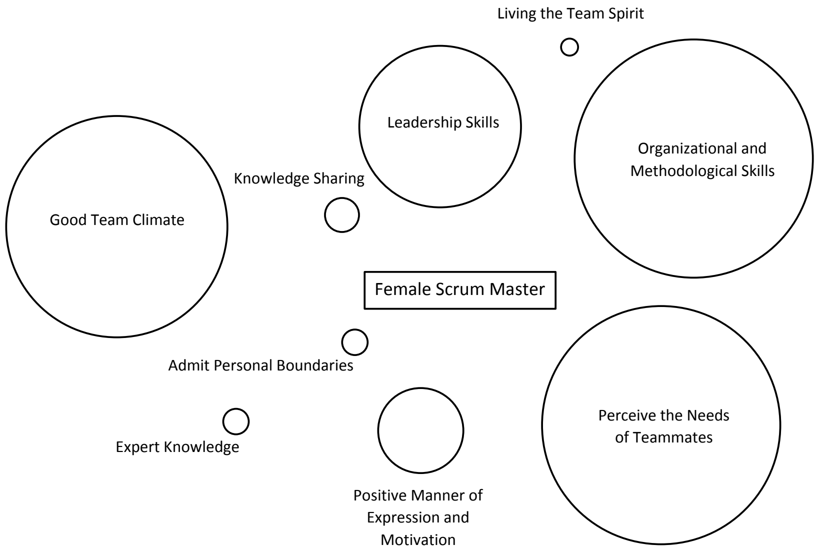


Figure 6.1.: Bubble chart illustrating the codes after axial coding of the transcripts of the weekly Scrum meetings for the team with a female Scrum master. The size of the bubbles represents the frequency of use of the respective code.

purpose.

Since this is an elective course which could be taken by students from different Bachelor study programs, participants were not mainly from Computer Science but from Information Management and Corporate Communications or Business Information Systems Engineering programs. Due to this fact, one third of the 11 participants were female, which is a comparably high percentage compared to usual Computer Science classes.

The participants formed two agile project teams with 5 and 6 students, respectively. Each group first had to create an initial product vision for the app they wanted to develop themselves. The only limitation was that the apps later should support their fellow students during their daily life at the university.

Both teams had to develop the mobile app natively for the Android platform[1], using Java as the programming language and the embedded HSQL database within Android. Development was done using the Eclipse IDE with the Android Developer Tool (ADT) Plugin[2].

The participants worked at their project for about 3 months using an adapted Scrum methodology. First, the teams organized themselves and chose their respective Scrum master or mistress. Both teams chose a female student as the Scrum master. The teacher took the role of the product owner. Both teams managed to do 3 sprints with a weekly (not daily) Scrum (stand-up) meeting. The Scrum meetings always took place at the beginning of the weekly lectures. Both teams implemented the app successfully and completed the assignment with a very good grade.

To evaluate the team behavior and the interaction of the participants, we conducted an exploratory qualitative analysis. The weekly Scrum meetings of one student group with the female Scrum master were videotaped for later analysis. In addition, at the end of the semester we handed out a questionnaire to the participants to receive feedback on the effectiveness of the Scrum process from their point of view. Additionally, we asked the students about their own opinion about how important they considered themselves for the team during the different phases of the project and why. Our raw data thus were the video footage and the filled out questionnaires.

For further evaluation, the video footage of the Scrum meetings was transcribed manually into textual form. This enabled us to apply principles of Grounded Theory [GS99] for the analysis of the transcripts. Therefore, we first we performed open coding followed by axial coding in a second step.

6.4. Results

The codes obtained from the axial coding of the transcripts are illustrated in the bubble chart of Figure 6.1. Each bubble represents one of the codes identified after axial coding. The diameter of the bubbles corresponds to the frequency of use of the respective code. However, it is important to emphasize that despite it is interesting to know how frequent a code was used, the reader has to keep in mind that the frequency does not correspond necessarily to the importance of the code. In Grounded Theory, every statement made can be important independent of the frequency.

[1]https://www.android.com/

[2]`http://marketplace.eclipse.org/content/android-development-tools-eclipse`

Figure 6.1 illustrates that in the video transcripts very often indications were observed for organizational and methodological skills of the female Scrum master. We found hints for perceiving the needs of teammates with the same frequency. Leadership skills were coded in a high frequency, too. All three of them together are good features of a Scrum master. In addition we found hints for some supplementary skills. Those were knowledge sharing, a positive manner of expression and motivating the team, living the team spirit and thus being a good example for the team, expert knowledge, and admitting personal boundaries.

We observed that this mixture of skills contributed to a good, respectful, communicative team climate. As Figure 6.1 shows, we very often found hints for that.

The female Scrum master showed *leadership skills*. She exhibited a certain degree of firmness and expressed her opinion self-confidently. She committed herself to clear decisions and she was able to assign tasks. In addition, she stood critique.

She answered technical questions and made technical statements. Her teammates asked her repeatedly technical questions which shows that they respected her knowledge. So with respect to *expert knowledge* she was a respected counterpart.

A good Scrum master needs to have *organizational and methodological skills*. The female Scrum master exhibited these skills. She reasonably structured and prioritized tasks. She structured the Scrum meetings, was focused on important topics and determined reasonable to-dos for the next sprint, sometimes by splitting them into smaller steps. She was able to specify problems with the help of questions and made constructive proposals.

However, all these skills could have been exhibited by a male Scrum master, too. The next results show the female specific strengths. The female Scrum master *perceived the needs of her teammates, respected them and was responsive to them*. She included the whole team in the decision process. She listened to problems and wishes and found solutions or included the wishes of the others in the next steps. She actively asked for the opinion of the others and for the needs of her teammates. E.g. one of the team members had an important date during the next sprint, so the Scrum master tried to find an appropriate solution for the whole group. And she actively offered help.

Another important virtue is to *share knowledge*. Our female Scrum master was open to share her knowledge with the whole team and included the team in decision-making processes. This confirms results of other researchers who found that women have a more positive attitude towards knowledge sharing than men (see e.g.[BIA06],

[CK03]).

The female Scrum master “lived” team spirit. She accredited (partial) results not to herself but to the whole team and expressed that verbally. Concerning tasks, she asked if every one agrees with the task allocation.

Another observation made while coding the transcripts was *the usage of positive words and active motivation of the team*. The female Scrum master formulated poor results in a non discriminating way so that the person affected still felt respected and encouraged to improve the delivered work.

To *admit personal boundaries to others* is something untypical for men. The female Scrum master had this ability. We had the impression that all in all this lead to a good climate in the team. The teammates felt free to tell if something did not work well. They felt free to utter their ideas or critique. Two other recognizable results were the active offer of help from the teammates and a constructive working atmosphere in the team.

Concerning the questionnaires, three students returned them filled out completely. Among them was the questionnaire of the observed female Scrum master. She evaluated the Scrum process the same way we did when we evaluated the video tapes. Her assessment was that at the beginning the Scrum process was ineffective, the team spent too much time on subsidiary topics. The effectiveness improved from meeting to meeting. Analyzing the video tapes we had the same impression.

Concerning the project phase she felt to be very important for the team, she answered it was during the beginning phase by motivating the team and in the end phase by structuring the team and allocating the tasks. These were exactly the things, we determined while coding the video transcripts. Her self-perception perfectly matches our external perception.

In summary, the analysis of the transcripts of the Scrum meetings provides a strong evidence for the positive impact of a female Scrum master and a typical “female” behavior on the performance and spirit of agile development teams. So it can be concluded that indeed a female Scrum aster might lead to better performing agile software development teams.

6.5. Limitations and Further Research

The strongest limitation is the nature of the performed study. So far, we have only performed an exploratory qualitative case study with one group of students in an agile student software project. Despite the fact that the observations made are conclusive and in agreement with previous results about gender impact on team performance, their validity needs to be analyzed further. Especially a direct comparison of team performance of an agile team with a male Scrum master is of high interest.

Therefore it is necessary to extend the present qualitative study to more participants as well as to repeat it with agile teams in real-world software projects to verify the observations made. In addition, a quantitative study should follow to obtain statistically significant results.

6.6. Conclusion

In conclusion, we have performed an exploratory qualitative case study within the context of a students' software project with two agile teams, each developing a smart phone app. In particular, the role and impact of a female Scrum master was analyzed by means of Grounded Theory.

Our results show that a female management style perfectly suits the role of a Scrum master. In addition to the "ordinary" skills which are required by any Scrum master in general, and can be performed by a male person, the female management style promotes team spirit and a constructive, communicative working atmosphere. A female Scrum master perceives the needs of her team members. She includes the whole team in decision processes and shares knowledge.

We believe that female Scrum masters fulfill this role even more successful than their male colleagues. Of course, our investigation is still strongly limited and of exploratory nature. However, we consider this question worth to be studied further. We are interested in a direct comparison of team performance of Scrum teams between two ore more teams with at least one female and one male Scrum master. This will be the first of our next steps. With the present Chapter we hope to encourage other researchers to intensify studies in this research area and find out whether our assumption is right.

We hope studies like this will help to bring more women and generally more diversity into the IT sector in the long run. And maybe Martin Fowler's vision will come true one day: "I have a different vision - one that sticks it to the suits so hard it will make their eyes water. How about a community where women are valued for their ability to program and not by the thickness of their skin? How about a community that edgily pushes new boundaries without reinforcing long running evils? Perhaps even a community where women reach equal numbers? Such a community would hand the suits the defeat in the long battle women have been fighting for centuries. I'd love to be part of that." [Fow09].

Chapter 7
Conclusion

Personality was found to be one significantly influencing factor e.g. on team performance [dFS+13; YOCC17]. This work explored the influence of personality on the quality of software – the outcome of a software engineering process which in most cases is team work.

7.1. Summary

To explore this connection, in Chapter 3 we first conducted a **systematic literature review** following the guidelines of Kitchenham and Charters [KC07] and examined how the impact of the personality of a software engineer on software quality can be described. We found that existing studies show a link between human factors or personality types or traits and software quality in general. But there were only two studies which mentioned an influence on distinct software quality aspects. We expected to find more such links but research conducted until now is far from being sufficient to provide a strong basis for making clear statements on the link between the personality of a software engineer and distinct software quality attributes. In particular, the literature review revealed the following **gaps**:

- Far too few investigations have been made in the industrial environment, most study subjects were students.

- The number of studied subjects was too small to make statistically significant statements.
- The validity and reliability of the personality type tests used were questionable.

For this reason, we carried out a quantitative and a qualitative study with experts from the field. In both studies we used a personality test that is both reliable and valid.

The quantitative study – which can be found in Chapter 4 – explored if a prediction of maintainability by personality is possible. Therefore we used the **Bayesian data analysis method**. With the help of this method we were able to generate a model which predicts 17 software maintainability metrics by personality traits of the HEXACO personality model [LA09a]. We also analyzed differences between four distinct personality traits and their influence on 17 software metrics and found that there are differences between distinct personality traits.

Beyond that, we provided practitioners with suggestions on how to optimally staff a software development team with regard to personality. Chapter 5 presents results of our qualitative study: we lead interviews with experts from the field and analyzed them following a **Grounded Theory approach**.

The literature review also revealed that – beyond personality – gender plays a role in the composition of high performing teams. Therefore we conducted a qualitative study that examined the role of gender in Scrum teams, especially the role of the Scrum Master (see Chapter 6).

7.2. Contributions

This work thus creates a contribution to closing the research gap by:

- giving a profound literature-based evaluation of research gaps in the field of influence of personality on software quality
- the studied subjects are experts from the field
- the used personality model has proven to be valid and reliable
- predicting maintainability metrics by personality traits of a software engineer
- providing a model for the above mentioned prediction
- proving the applicability and showing benefits of the Bayesian data analysis method for research in this area

- providing practitioners with suggestions how to staff different software engineering roles with respect to personality
- providing practitioners with research results on the staffing of the scrum master role.

7.3. Limitations

The main limitation of my work is the low number of study participants in Chapter 4. Despite years of acquisition it was not possible to win more participants in Germany. Maybe the studies of this dissertation could be extended in other countries where the mentality regarding data processing is different than in Germany.

Another limitation is that I worked alone for the majority of the data evaluation. In order to maintain the quality of the systematic literature review as well as the qualitative studies, I called in other researchers who evaluated random parts of the collected data and compared their results with mine. If the agreement was too low, the evaluation scheme was improved.

In general, I conducted my studies in Germany. It should be investigated whether the study results would be the same if the studies had taken place in other countries.

My research only takes into account the personality profile of a person. Other factors, such as cognitive abilities, should also be taken into account in further investigations as possible additional influencing factors.

7.4. Recommendations

It is possible to build a model which predicts maintainability from personality. For that Bayesian data analysis is a powerful method. Not only because this method enables predictions but also because it is not limited to the acceptance or rejection of hypotheses. With the help of Bayesian data analysis, differences between groups can be investigated and quantitative statements can be made about the extent of differences between those different groups. Therefore, we recommend this method to those who do not want to test hypotheses but are interested in quantitative differences between groups.

Roles in a software development team should be staffed with respect to personality traits in order to build high performing teams. And it is advantageous to fill the Scrum Master role with a female person.

There should always be a parallel ethical discussion about the meaningfulness and consequences of investigations and also developments in the field of software and software engineering. Chapter 5 is an example of this. Is it ethically and morally justifiable to be "only" guided by a personality profile when filling vacancies? This is already the practice in many large companies today. But doesn't this reduce the person to a small part of him or her?

The same applies to the development of software. Sometimes possibilities are implemented – i.e. software is programmed – without thinking about the consequences. Therefore, a recommendation as a consequence of the research for this dissertation is that software development teams should always include humanities or social scientists who study the ethical moral implications of a software component. Current software development teams are put together primarily with a view to ensuring that software can be created technically in high quality. However, the "social quality" of software is not yet taken into account.

7.5. Future Research

During my research, I naturally came up against limits, but at the same time these limits offer suggestions for further research. These I list in the following:

- Further studies should be conducted among experts from different branches of industry.
- Studies should be conducted in different countries and results should be compared.
- Studies could concentrate on particular roles of people in a software engineering process, e.g. architects, programmers, or testers.
- As mentioned above, a valid and reliable personality type test should be used for the investigations. Additionally, studies could concentrate on one aspect of software product quality and should clearly define the metrics that were used to measure the distinct aspect of software product quality. Having all these data, researchers could investigate all kinds of correlations between personality type and different aspects of software product quality.
- Studies should incorporate a larger number of participants. This would allow causal models to be established and investigated.
- There should be joint research between software engineering and computer

science scientists and humanities or social scientists. The field of research here is very broad.

- Metrics for the “social quality” of software should be established.

Bibliography

[19] *NATO Software Engineering Conference 1968*. 2019. URL: `http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/` (visited on 04/08/2019) (cit. on p. 18).

[AA12] A. April, A. Abran. *Software maintenance management: evaluation and continuous improvement*. Vol. 67. John Wiley & Sons, 2012 (cit. on pp. 38, 40, 41).

[AAI12] J. Apesteguia, G. Azmat, N. Iriberri. 'The Impact of Gender Composition on Team Performance and Decision Making: Evidence from the Field'. In: *Management Science* 58.1 (2012), pp. 78–93 (cit. on pp. 146, 149).

[ABA11] M. André, M. G. Baldoquín, S. T. Acuña. 'Formal model for assigning human resources to teams in software projects'. In: *Information and Software Technology* 53.3 (2011), pp. 259–275 (cit. on p. 127).

[ACCA11] N. D. Anh, D. S. Cruzes, R. Conradi, C. Ayala. 'Empirical validation of human factors in predicting issue lead time in open source projects'. In: *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*. ACM, 2011, pp. 1–10 (cit. on pp. 72, 247).

[AGE95] F. B. Abreu, M. Goulão, R. Esteves. 'Toward the Design Quality Evaluation of Object-Oriented Software Systems'. In: *Proceedings of the 5th International Conference on Software Quality*. 1995 (cit. on p. 66).

[AGH+15] S. T. Acuña, M. N. Gómez, J. E. Hannay, N. Juristo, D. Pfahl. 'Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment'. In: *Information and Software Technology* 57 (2015), pp. 141–156 (cit. on p. 124).

[AGJ09] S. T. Acuña, M. Gómez, N. Juristo. 'How do personality, team processes and task characteristics relate to job satisfaction and software quality?' In: *Information and Software Technology* 51.3 (2009), pp. 627–639. URL: `http://www.sciencedirect.com/science/article/pii/S0950584908001080` (cit. on pp. 19, 72, 85, 239).

[Ahl18] A. Ahlers. 'Die Persönlichkeitsentwicklung ist niemals fertig'. In: *Zeit online* (2018). URL: `https://www.zeit.de/wissen/2018-05/psychologie-persoenlichkeit-entwicklung-alter-forschung/seite-2` (visited on 04/04/2019) (cit. on pp. 20, 81).

[AJ01] Alistair Cockburn, Jim Highsmith. 'Agile software development: the people factor - Computer'. In: *Computer* 34.11 (2001), pp. 131–133. URL: `http://ieeexplore.ieee.org/ielx5/2/20799/00963450.pdf?tp=&arnumber=963450&isnumber=20799` (visited on 12/06/2012) (cit. on pp. 149, 150).

[AJM06] S. Acuna, N. Juristo, A. Moreno. 'Emphasizing human capabilities in software development'. In: *IEEE Software* 23.2 (2006), pp. 94–101 (cit. on pp. 20, 67–70).

[AL07] M. C. Ashton, K. Lee. 'Empirical, theoretical, and practical advantages of the HEXACO model of personality structure'. In: *Personality and social psychology review : an official journal of the Society for Personality and Social Psychology, Inc* 11.2 (2007), pp. 150–166 (cit. on p. 126).

[AL08] M. C. Ashton, K. Lee. 'The prediction of Honesty–Humility-related criteria by the HEXACO and Five-Factor Models of personality'. In: *Journal of Research in Personality* 42.5 (2008), pp. 1216–1228 (cit. on p. 126).

[ALG04] M. C. Ashton, K. Lee, L. R. Goldberg. 'A Hierarchical Analysis of 1,710 English Personality-Descriptive Adjectives'. In: *Journal of Personality and Social Psychology* 87.5 (2004), pp. 707–721. URL: `https://search.proquest.com/docview/614397034?accountid=15918` (cit. on p. 134).

[All] N. L. Allison. 'The relationship between personality characteristics and job satisfaction of selected computer programmers'. Doctoral Thesis. Oklahoma State University. URL: `https://shareok.org/`

bitstream/handle/11244/19282/Thesis-1983D-A439r.pdf?sequence=1 (visited on 04/11/2019) (cit. on p. 19).

[ALP+04] M. C. Ashton, K. Lee, M. Perugini, P. Szarota, R. E. de Vries, L. Di Blas, K. Boies, B. de Raad. 'A Six-Factor Structure of Personality-Descriptive Adjectives: Solutions From Psycholexical Studies in Seven Languages'. In: *Journal of Personality and Social Psychology* 86.2 (2004), pp. 356–366. URL: https://search.proquest.com/docview/614386182?accountid=15918 (cit. on p. 134).

[And19] Andrew J. Ko. *The history of software engineering*. 2019. URL: https://faculty.washington.edu/ajko/books/cooperative-software-development/history.html (visited on 04/08/2019) (cit. on pp. 17, 18).

[AO15] N. J. Allen, T. O'Neill. 'Team Composition and Performance'. In: *The psychology and management of project teams: An interdisciplinary perspective*. Ed. by F. Chiocchio, E. K. Kelloway, B. Hobbs. Oxford and New York, NY: Oxford University Press, 2015, pp. 301–328 (cit. on pp. 126, 146).

[ARBH15] A. Amin, M. Rehman, S. Basri, M. F. Hassan. 'A proposed conceptual framework of programmer's creativity'. In: *2015 International Symposium on Technology Management and Emerging Technologies (ISTMET): 25 - 27 Aug. 2015, Langkawi Island, Malaysia*. Piscataway, NJ: IEEE, 2015, pp. 108–113 (cit. on p. 143).

[AS95] E. Aas, I. Sundsbo. 'Harnessing the human factor for quality designs'. In: *IEEE Circuits and Devices Magazine* 11.3 (1995), pp. 24–28 (cit. on p. 73).

[Ash18] M. C. Ashton. *Individual differences and personality*. Third edition. London et al.: Academic Press is an imprint of Elsevier Ltd, 2018 (cit. on pp. 26–31, 63, 87, 125, 127, 128, 134, 145).

[BBHS] S. Beecham, N. Baddoo, T. Hall, H. Sharp. *Protocol for a Systematic Literature Review of Motivation in Software Engineering*. Ed. by University of Hertfordshire. URL: http://uhra.herts.ac.uk/handle/2299/992 (cit. on p. 59).

[BBL76] B. W. Boehm, J. R. Brown, M. Lipow. 'Quantitative Evaluation of Software Quality'. In: *Proceedings of the 2Nd International Conference on Software Engineering*. ICSE '76. Los Alamitos, CA, USA: IEEE Computer Society Press, 1976, pp. 592–605. URL: http://dl.acm.org/citation.cfm?id=800253.807736 (cit. on p. 33).

[BBM96] V. R. Basili, L. C. Briand, W. L. Melo. 'A validation of object-oriented design metrics as quality indicators'. In: *IEEE Transactions on Software Engineering* 22.10 (1996), pp. 751–761 (cit. on p. 19).

[BC01] S. Burke, K. M. Collins. 'Gender differences in leadership styles and management skills'. In: *Women in Management Review* 16.5 (2001), pp. 244–257. URL: https://www.emerald.com/insight/content/doi/10.1108/09649420110395728/full/pdf?journalCode=wimrUR%20-%20https://www.emerald.com/insight/content/doi/10.1108/09649420110395728/full/html?journalCode=wimr (cit. on p. 151).

[BD14] D. Bishop, A. Deokar. 'Toward an Understanding of Preference for Agile Software Development Methods from a Personality Theory Perspective'. In: *IEEE 8th International Symposium on Service-Oriented System Engineering (SOSE), 2014: 7 - 11 April 2014, Oxford, United Kingdom ; [including workshop/simposium papers]*. Piscataway, NJ: IEEE, 2014, pp. 4749–4758 (cit. on p. 74).

[Bel10] R. M. Belbin. *Team roles at work*. 2. ed. Amsterdam: Butterworth-Heinemann, 2010. URL: http://www.sciencedirect.com/science/book/9781856178006 (cit. on pp. 127, 145).

[Ber08] B. Berenbach. 'The other skills of the software architect'. In: *Proceedings of the first international workshop on Leadership and management in software architecture*. ACM, 2008, pp. 7–12 (cit. on pp. 68, 69, 206).

[BFHD07] L. Blum, C. Frieze, O. Hazzan, M. B. Dias. 'A Cultural Perspective on Gender Diversity in Computing'. In: (2007) (cit. on p. 150).

[BH] A. Benlian, T. Hess. *Does personality matter in the evaluation of ERP systems? Findings from a conjoint study with IS purchasing managers*. URL: http://ideas.repec.org/p/dar/wpaper/57928.html (cit. on pp. 67, 206).

[BIA06] P. Bordia, B. E. Irmer, D. Abusah. 'Differences in sharing knowledge interpersonally and via databases: The role of evaluation apprehension and perceived benefits'. In: *European Journal of Work and Organizational Psychology* 15.3 (2006), pp. 262–280 (cit. on p. 154).

[Bis95] C. Bishop-Clark. 'Cognitive style, personality, and computer programming'. In: *Computers in Human Behavior* 11.2 (1995), pp. 241–260 (cit. on pp. 69, 70, 203).

[BIT] BITKOM. *In Deutschland fehlen 41.000 IT-Experten*. URL: `https://www.bitkom.org/de/markt%5Ctextunderscore%20statistik/64054%5Ctextunderscore%2080733.aspx` (visited on 03/27/2015) (cit. on p. 148).

[BIT14] BITKOM. *IT-Spezialistinnen werden dringend gesucht (2014)*. 2014. URL: `https://www.bitkom.org/de/presse/81149%5Ctextunderscore%2079021.aspx` (visited on 03/30/2015) (cit. on p. 148).

[BK81] R. P. Bostrom, K. M. Kaiser. 'Personality differences within systems project teams: Implications for designing solving centers'. In: *Proceedings of the eighteenth annual computer personnel research conference*. ACM, 1981, pp. 248–285 (cit. on p. 203).

[BKPB13] B. H. Bradley, A. C. Klotz, B. E. Postlethwaite, K. G. Brown. 'Ready to rumble: how team personality composition and task conflict interact to improve performance'. In: *The Journal of applied psychology* 98.2 (2013), pp. 385–392 (cit. on p. 124).

[BLK+07] F. Belanger, T. Lewis, G. M. Kasper, W. J. Smith, K. V. Harrington. 'Are Computing Students Different? An Analysis of Coping Strategies and Emotional Intelligence'. In: *IEEE Transactions on Education* 50.3 (2007), pp. 188–196 (cit. on p. 73).

[BM96] F. Brito e Abreu, W. Melo. 'Evaluating the impact of object-oriented design on software quality'. In: *Proceedings of the 3rd International Software Metrics Symposium: March 25-26, 1996, Berlin, Germany*. Los Alamitos, Calif.: IEEE Computer Society Press, 1996, pp. 90–99 (cit. on pp. 47, 66).

[BOJH08] M. Baer, G. R. Oldham, G. C. Jacobsohn, A. B. Hollingshead. 'The Personality Composition of Teams and Creativity: The Moderating Role of Team Creative Confidence'. In: *Journal of Creative Behavior* 42.4 (2008), pp. 255–282 (cit. on p. 125).

[BPW02] P. Balthazard, R. Potter, J. Warren. 'The effects of extraversion and expertise on virtual team interaction and performance'. In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. 2002, p. 10 (cit. on p. 218).

[BPW04] P. Balthazard, R. E. Potter, J. Warren. 'Expertise, extraversion and group interaction styles as performance indicators in virtual teams: how do perceptions of IT's performance get formed?' In: *SIGMIS Database* 35.1 (2004), pp. 41–64 (cit. on p. 221).

[BR00] K. H. Bennett, V. T. Rajlich. 'Software maintenance and evolution: a roadmap'. In: *Proceedings of the Conference on The Future of Software Engineering*. Ed. by A. Finkelstein. 2000, pp. 73–87 (cit. on p. 40).

[Bry16] A. Bryman. *Social Research Methods*. 5th ed. Oxford University Press Oxford, 2016. URL: https://books.google.de/books?id=vCq5m2hPkOMC (cit. on pp. 129, 131, 132).

[BS48] K. D. Benne, P. Sheats. 'Functional Roles of Group Members'. In: *Journal of Social Issues* 4.2 (1948), pp. 41–49 (cit. on p. 223).

[BSB08] C. Bommer, M. Spindler, V. Barr. *Softwarewartung: Grundlagen, Management und Wartungstechniken*. 1. Aufl. Heidelberg: Dpunkt-Verl., 2008 (cit. on pp. 32, 38–41, 86).

[BSNM98] M. R. Barrick, G. L. Stewart, M. J. Neubert, M. K. Mount. 'Relating member ability and personality to work-team processes and team effectiveness'. In: *Journal of Applied Psychology* 83.3 (1998), pp. 377–391 (cit. on p. 126).

[BT] J. D. Bedingfield, A. E. Thal. 'Project manager personality as a factor for success'. In: *Proceedings of the Portland International Conference on Management of Engineering & Technology 2008*, pp. 1303–1314 (cit. on pp. 67, 205).

[BW11] J. B. Bear, A. W. Woolley. 'The role of gender in team collaboration and performance'. In: *Interdisciplinary science reviews* 36.2 (2011), pp. 146–153 (cit. on p. 149).

[BZG05] G. Beranek, W. Zuser, T. Grechenig. 'Functional group roles in software engineering teams'. In: *SIGSOFT Softw. Eng. Notes* 30.4 (2005), pp. 1–7 (cit. on pp. 151, 223).

[CA10a] L. Capretz, F. Ahmed. 'Making Sense of Software Development and Personality Types'. In: *IT Professional* 12.1 (2010), pp. 6–13 (cit. on pp. 20, 125).

[CA10b] L. F. Capretz, F. Ahmed. 'Why do we need personality diversity in software engineering?' In: *SIGSOFT Softw. Eng. Notes* 35.2 (2010), pp. 1–11 (cit. on pp. 68–70, 207).

[Cap03] L. F. Capretz. 'Personality types in software engineering'. In: *International Journal of Human-Computer Studies* 58.2 (2003), pp. 207–214 (cit. on p. 214).

[CdM+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: *Proceedings of the 15th Annual Conference on Evaluation & Assessment in Software Engineering*. 2011, pp. 1–10 (cit. on pp. 54, 246).

[CdSM12] B. Crawford, de La Barra, Claudio Leon, R. Soto, E. Monfroy. 'Agile software teams must be creatives'. In: *Proceedings of the Workshop on Engineering Applications 2012*. 2012, pp. 1–6 (cit. on p. 207).

[CH06] C. G. Cegielski, D. J. Hall. 'What makes a good programmer?' In: *Communications of the ACM* 49.10 (2006), pp. 73–75 (cit. on pp. 73, 237).

[CH09] J.-C. Chen, S.-J. Huang. 'An empirical analysis of the impact of software development problem factors on software maintainability'. In: *Journal of Systems and Software* 82.6 (2009), pp. 981–992 (cit. on p. 33).

[Cha05] R. N. Charette. 'Why Software Fails: We waste billions of dollars each year on entirely preventable mistakes'. In: *IEEE Spectrum* (2005). URL: `https://spectrum.ieee.org/computing/software/why-software-fails` (visited on 07/04/2018) (cit. on p. 53).

[CK03] C. E. Connelly, E. Kevin Kelloway. 'Predictors of employees' perceptions of knowledge sharing cultures'. In: *Leadership & Organization Development Journal* 24.5 (2003), pp. 294–301 (cit. on p. 155).

[CK91] S. R. Chidamber, C. F. Kemerer. 'Towards a metrics suite for object oriented design'. In: *OOPSLA '91: Object-oriented programming systems, languages, and applications : conference proceedings*. Ed. by A. Paepcke. Special issue of SIGPLAN notices. New York: ACM Press, 1991, pp. 197–211 (cit. on p. 37).

[Cla17] Claudia Tödtmann. *Gallup-Studie: Vorgesetzte schädigen die Firma, wenn Sie das Thema Führung nicht beherrschen*. Ed. by Handelsblatt GmbH. 2017. URL: `http://blog.wiwo.de/management/2017/03/22/gallup-studie-vorgesetzte-schaedigen-die-firma-wenn-sie-das-thema-fuehrung-nicht-beherrschen/` (visited on 03/06/2018) (cit. on p. 144).

[CMH04] F. Coffield, D. Moseley, Hall, Elaine, Ecclestone, Kathryn. 'Learning styles and pedagogy in post-16 learning: A systematic and critical review'. In: *learning&skills research centre* (2004). URL: `https://web.archive.org/web/20081205043419if_/http://www.lsda.org.uk/files/PDF/1543.pdf` (visited on 08/17/2018) (cit. on pp. 127, 128).

[CMP92] P. T. Costa, R. R. McCrae, I. Psychological Assessment Resources. *Revised NEO Personality Inventory (NEO PI-R) and NEO Five-Factor Inventory (NEO-FFI)*. Psychological Assessment Resources, 1992. URL: `https://books.google.de/books?id=mp3zNwAACAAJ` (cit. on pp. 29, 126).

[CPT+19] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, F. Ferrucci. 'Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?' In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software engineering in society*. Piscataway, NJ: IEEE, 2019, pp. 11–20 (cit. on p. 148).

[CS15] J. M. Corbin, A. L. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Fourth edition. Los Angeles et al.: SAGE, 2015 (cit. on pp. 130, 131).

[CSF14] N. Christiansen, M. Sliter, C. T. Frost. 'What employees dislike about their jobs: Relationship between personality-based fit and work satisfaction'. In: *Personality and Individual Differences* 71 (2014), pp. 25–29 (cit. on p. 74).

[CSSV15] A. Chagas, M. Santos, C. Santana, A. Vasconcelos. 'The Impact of Human Factors on Agile Projects'. In: *2015 Agile Conference (AGILE 2015)*. Piscataway, NJ: IEEE, 2015, pp. 87–91 (cit. on p. 150).

[CT98] D. Carrington, J. Teague. 'Personality type, career preference and implications for computer science recruitment and teaching'. In: *Proceedings of the 3rd Australasian conference on Computer science education ACSE '98*. 1998, pp. 155–163 (cit. on pp. 68, 69, 204).

[CWW03] J. G. Clark, D. B. Walz, J. L. Wynekoop. 'Identifying exceptional application software developers: A comparison of students and professionals'. In: *Communications of the Association for Information Systems* 11.1 (2003), Article 8 (cit. on p. 234).

[Dan13] Daniel Graziotin. 'The Dynamics of Creativity in Software Development'. In: *Proceedings of the PROFES 2013*. 2013, pp. 1–6. URL: `https://arxiv.org/ftp/arxiv/papers/1305/1305.6045.pdf` (visited on 05/17/2018) (cit. on p. 143).

[Den16] S. Denning. 'How to make the whole organization "Agile"'. In: *Strategy & Leadership* 44.4 (2016), pp. 10–17 (cit. on p. 74).

[dFS+13] F. Q. da Silva, A. C. C. França, M. Suassuna, de Sousa Mariz, Leila M.R., I. Rossiley, R. C. de Miranda, T. B. Gouveia, C. V. Monteiro, E. Lucena, E. S. Cardozo, E. Espindola. 'Team building criteria in software projects: A mix-method replicated study'. In: *Information and Software Technology* 55.7 (2013), pp. 1316–1340 (cit. on pp. 19, 126, 159).

[DG07] A. D. Da Cunha, D. Greathead. 'Does personality matter?: an analysis of code-review ability'. In: *Communications of the ACM* 50.5 (2007), pp. 109–112 (cit. on pp. 70, 72, 205, 238).

[DG12] E. G. Daylight, K. de Grave. *The dawn of software engineering: From Turing to Dijkstra*. Heverlee: Lonely Scholar, 2012 (cit. on p. 17).

[Dij72] E. W. Dijkstra. ‘The Humble Programmer’. In: *Commun. ACM* 15.10 (1972), pp. 859–866. URL: http://doi.acm.org/10.1145/355604.361591,%20Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen (cit. on p. 18).

[Dir13] Dirk W. Hoffmann. *Software-Qualität*. 2nd ed. Berlin, Heidelberg: Springer Vieweg, 2013 (cit. on p. 35).

[DL14] T. DeMarco, T. R. Lister. *Peopleware: Productive projects and teams*. Third edition. Upper Saddle River, NJ: Addison-Wesley, 2014 (cit. on p. 127).

[DM05] D. Darcy, Meng Ma. ‘Exploring Individual Characteristics and Programming Performance: Implications for Programmer Selection’. In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. 2005, 314a (cit. on pp. 19, 71, 73, 85, 235).

[DP08] Dr. Rüdiger Lincke, Prof. Dr. Welf Löwe. *VizzMaintenance*. 2008. URL: http://www.arisa.se/products.php (cit. on pp. 117–119).

[DR01] Daniel Rodriguez, Rachel Harrison. *An Overview of Object-Oriented Design Metrics*. 2001 (cit. on p. 47).

[DRG10] A. Dearden, H. Rizvi, S. Gupta. ‘Roles and Responsibilities in Agile ICT for Development’. In: *Proceedings of the 2010 International Conference on Interaction Design*. IHCI’10. Mumbai, India: British Computer Society, 2010, pp. 19–30. URL: http://dl.acm.org/citation.cfm?id=2227347.2227350 (cit. on p. 150).

[DSZY11] Deng Rui, She Wei, Zhao Li, Yang Hong. ‘A study on the cognitive style of software developers’. In: *ACM Inroads* 2.3 (2011), pp. 43–46 (cit. on p. 207).

[DWP+] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, J.-F. Girard. ‘An Activity-Based Quality Model for Maintainability’. In: *2007 IEEE International Conference on Software Maintenance*, pp. 184–193 (cit. on pp. 33–36, 38, 41).

[EWIM11] A. W. R. Emanuel, R. Wardoyo, J. E. Istiyanto, K. Mustofa. ‘Statistical Analysis on Software Metrics Affecting Modularity in Open Source Software’. In: *International Journal of Computer Science & Information Technology* 3.3 (2011), pp. 35–50 (cit. on p. 46).

[FB15] N. E. Fenton, J. Bieman. *Software Metrics: A Rigorous and Practical Approach*. Chapman et Hall/CRC innovations in software engineering and software development. Boca Raton, Florida: CRC Press, 2015 (cit. on pp. 32, 33, 46, 47).

[FB99] M. Fowler, K. Beck. *Refactoring: Improving the design of existing code*. The Addison-Wesley object technology series. Reading, MA: Addison-Wesley, 1999 (cit. on p. 46).

[Fie13] A. Field. *Discovering statistics using IBM SPSS statistics: And sex and drugs and rock 'n' roll*. 4th edition. MobileStudy. Los Angeles et al.: SAGE, 2013 (cit. on p. 91).

[FL14] N. N. V. Ferreira, J. J. Langerman. 'The correlation between personality type and individual performance on an ICT Project'. In: *9th International Conference on Computer Science & Education (ICCSE), 2014: 22 - 24 Aug. 2014, Vancouver, Canada*. Piscataway, NJ: IEEE, 2014, pp. 425–430 (cit. on p. 216).

[Fli17] U. Flick. *Qualitative Sozialforschung: Eine Einführung*. 8. Auflage, Originalausgabe. Vol. 55694. Rororo Rowohlts Enzyklopädie. Reinbek bei Hamburg: rowohlts enzyklopädie im Rowohlt Taschenbuch Verlag, 2017 (cit. on p. 129).

[FN01] G. D. Fenwick, D. J. Neal. 'Effect of Gender Composition on Group Performance'. In: *Gender, Work and Organization* 8.2 (2001), pp. 205–225 (cit. on p. 146).

[FN99] N. E. Fenton, M. Neil. 'Software metrics: successes, failures and new directions'. In: *Journal of Systems and Software* 47.2 (1999), pp. 149–157. URL: `http://www.sciencedirect.com/science/article/pii/S0164121299000357` (cit. on p. 37).

[Fow09] M. Fowler. *SmutOnRails*. 2009. URL: `https://martinfowler.com/bliki/SmutOnRails.html` (visited on 12/05/2020) (cit. on pp. 148, 157).

[FWW98] S. G. Fisher, K. W.D., J. Wong. 'Cognitive style and team role preference'. In: *Journal of Managerial Psychology* 13.8 (1998), pp. 544–557 (cit. on p. 124).

[Gal04] M. J. Gallivan. 'Examining IT professionals' adaptation to technological change: the influence of gender and personal attributes'. In: *SIGMIS Database* 35.3 (2004), pp. 28–49 (cit. on pp. 71, 73).

[Gar96] W. Gardner. 'Using the Myers-Briggs Type Indicator to study managers: A literature review and research agenda'. In: *Journal of Management* 22.1 (1996), pp. 45–83. URL: `https://ac.els-cdn.com/S0149206396900124/1-s2.0-S0149206396900124-main.pdf?_tid=52c20eea-b1ba-4479-82f0-0dc85130d72e&acdnat=1528279339_0042a77b79121c1b90ad62bfb5d27c24` (visited on 06/06/2018) (cit. on pp. 127, 128).

[GB10] D. Gonçalves, J. Britz. 'Screening candidate systems engineers: exploratory results.' In: *Proceedings of the CSIR 3rd Biennial Conference 2010*. 2010, p. 14 (cit. on p. 73).

[GCF98] P. J. Guinan, J. G. Cooprider, S. Faraj. 'Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach'. In: *Info. Sys. Research* 9.2 (Feb. 1998), pp. 101–125. URL: `http://dx.doi.org/10.1287/isre.9.2.101` (cit. on p. 149).

[GCS+14] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, D. B. Rubin. *Bayesian data analysis*. Third edition. Texts in statistical science series. Boca Raton, London, and New York: CRC Press Taylor and Francis Group, 2014. URL: `http://gbv.eblib.com/patron/FullRecord.aspx?p=1438153` (cit. on p. 92).

[GFRV14] D. Grams, T. Frank, S. Rehberger, B. Vogel-Heuser. 'Female characteristics and requirements in software engineering in mechanical engineering'. In: *2014 International Conference on Interactive Collaborative Learning (ICL): 3 - 6 Dec. 2014, Dubai, UAE*. Piscataway, NJ: IEEE, 2014, pp. 272–279 (cit. on pp. 146, 248).

[GJO+16] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, A. Waqas. 'A rule-based model for software development team composition: Team leader role with personality types and gender classification'. In: *Information and Software Technology* 74 (2016), pp. 105–113 (cit. on p. 146).

[GJOT14] A. R. Gila, J. Jaafa, M. Omar, M. Z. Tunio. 'Impact of personality and gender diversity on software development teams' performance'. In: *International Conference on Computer, Communications, and Control Technology (I4CT), 2014: 2 - 4 Sept. 2014, Fave Hotel, Langkawi, Kedah, Malaysia*. Piscataway, NJ: IEEE, 2014, pp. 261–265 (cit. on p. 146).

[GL04] N. Gorla, Y. W. Lam. 'Who should work with whom?: building effective software project teams'. In: *Communications of the ACM* 47.6 (2004), pp. 79–82 (cit. on pp. 125, 223).

[Gla01] R. L. Glass. 'Frequently Forgotten Fundamental Facts About Software Engineering'. In: *IEEE Software* 18.3 (2001), pp. 112–111 (cit. on pp. 55, 84).

[GPM04] M. Genero, M. Piatini, E. Manso. 'Finding "early" indicators of UML class diagrams understandability and modifiability'. In: *Proceedings / 2004 International Symposium on Empirical Software Engineering, ISESE 2004: 19 - 20 August 2004, Redondo Beach, California*. Los Alamitos, Calif. [u.a.] and Los Alamitos, Calif. [u.a.]: IEEE Computer Society, 2004, pp. 207–216 (cit. on p. 48).

[GS10] B. G. Glaser, A. L. Strauss. *Grounded theory: Strategien qualitativer Forschung*. 3., unveränderte Auflage. Programmbereich Gesundheit. Bern: Verlag Hans Huber, 2010 (cit. on pp. 21, 131).

[GS99] B. G. Glaser, A. L. Strauss. *THE DISCOVERY OF GROUNDED THEORY: Strategies for Qualitative Research*. Aldine Pub., 1999 (cit. on p. 153).

[GT03] P. Grubb, A. A. Takang. *Software maintenance: concepts and practice*. World Scientific, 2003 (cit. on pp. 39, 40).

[GZG+14] T. J. Gandomani, H. Zulzalil, A. A. A. Ghani, A. B. M. Sultan, K. Y. Sharif. 'How Human Aspects Impress Agile Software Development Transition and Adoption'. In: *International Journal of Software Engineering and Its Applications* 8.1 (2014), pp. 129–148 (cit. on p. 86).

[H17] T. Hanna, h. online heise. *Programmiersprachen-Ranking: JavaScript vorne, Ruby mit Abwärtstrend*. 2017. URL: `https://www.heise.de/developer/meldung/Programmiersprachen-Ranking-JavaScript-vorne-Ruby-mit-Abwaertstrend-3739453.html` (visited on 07/15/2019) (cit. on p. 89).

[Hag98] Hagberg Consulting Group. *Career : Women and the Glass Ceiling*. 1998. URL: http://www.leader-values.com/article.php?aid=40 (visited on 03/30/2015) (cit. on p. 150).

[Han10] S. Hanenberg. 'Faith, hope, and love: an essay on software science's neglect of human factors'. In: *SIGPLAN Not* 45.10 (2010), pp. 933–946 (cit. on p. 54).

[HD06] O. Hazzan, Y. Dubinsky. 'Can Diversity in Global Software Development Be Enhanced by Agile Software Development?' In: *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*. GSD '06. Shanghai, China: ACM, 2006, pp. 58–61. URL: http://doi.acm.org/10.1145/1138506.1138520 (cit. on p. 150).

[HG08] J. P. T. Higgins, S. Green. *Cochrane Handbook for Systematic Reviews of Interventions*. Cochrane book series. Chichester, England, and Hoboken, NJ: Wiley-Blackwell, 2008 (cit. on p. 54).

[HKV07] I. Heitlager, T. Kuipers, J. Visser. 'A Practical Model for Measuring Maintainability'. In: *QUATIC 2007: 6th International Conference on the Quality of Information and Communications Technology : proceedings : 12-14 September, 2007, Lisbon, Portgual*. Ed. by R. J. Machado, F. B. e. Abreu, P. Rupino da Cunha. Los Alamitos, Calif.: IEEE Computer Society, 2007, pp. 30–39 (cit. on pp. 46, 47).

[HLW03] J. Hunsley, Lee C. M., Wood J. M. 'Controversial and Questionable Assessment Techniques'. In: *Science and Pseudoscience in Clinical Psychology*. Ed. by S. O. Lilienfeld, S. J. Lynn, J. M. Lohr. New York: Guilford Press, 2003, pp. 39–76 (cit. on pp. 27, 63, 81).

[HNB04] L. von Hellens, S. Nielsen, J. Beekhuyzen. 'An Exploration of Dualisms in Female Perceptions of IT Work'. In: *Journal of Information Technology Education* 3 (2004), pp. 104–116 (cit. on p. 149).

[Hog07] R. Hogan. *Personality and the fate of organizations*. Mahwah, NJ: Erlbaum, 2007. URL: http://www.loc.gov/catdir/enhancements/fy0662/2006010473-b.html (cit. on pp. 127, 128).

[Hon18] Honeypot, ed. *Frauen in der IT-Branche 2018 | Honeypot*. 2018. URL: https://www.honeypot.io/de/women-in-tech-2018/ (visited on 12/03/2020) (cit. on p. 148).

[Hop86] H. P. Hoplin. ‘Trends in computing careers: human factors in MIS’. In: *Proceedings of the twenty-second annual computer personnel research conference on Computer personnel research conference*. Calgary and Canada: ACM, 1986, pp. 143–150 (cit. on p. 54).

[HOv13] S. Hoogendoorn, H. Oosterbeek, M. van Praag. ‘The Impact of Gender Diversity on the Performance of Business Teams: Evidence from a Field Experiment’. In: *Management Science* 59.7 (2013), pp. 1514–1528 (cit. on pp. 146, 149).

[HT14] A. Hanif, S. Tariq. ‘An evaluation of personal and interpersonal competencies of project managers’. In: *2014 International Conference on Emerging Technologies (ICET): 8 - 9 Dec. 2014, Islamabad, Pakistan*. Piscataway, NJ: IEEE, 2014, pp. 19–23 (cit. on p. 208).

[htt20] https://www.apa.org. *Personality*. 2020. URL: `https://www.apa.org/topics/personality` (visited on 12/06/2020) (cit. on p. 26).

[IEE14] IEEE Computer Society. *Guide to the Software Engineering Body of Knowledge: Version 3.0*. Ed. by P. Bourque, R. E. Fairley. 2014. URL: `www.swebok.org` (cit. on pp. 38–40).

[IK11] R. Ivanova-Stenzel, D. Kübler. ‘Gender differences in team work and team competition’. In: *Journal of Economic Psychology* 32.5 (2011), pp. 797–808 (cit. on p. 149).

[IMI10] J. Iivonen, M. V. Mäntylä, J. Itkonen. ‘Characteristics of High Performing Testers: A Case Study’. In: *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2010, Article No. 60 (cit. on pp. 19, 72, 85, 242).

[iso17] iso25000.com. *ISO/IEC 25010*. Ed. by iso25000.com. 2017. URL: `http://iso25000.com/index.php/en/iso-25000-standards/iso-25010` (visited on 08/01/2017) (cit. on pp. 32, 34, 37).

[JBFe12] E. Janke, A. Bartel, P. Figas, et al. ‘Die Lehre von Software Engineering – eine Erhebung aus der Praxis’. In: *Tagungsband Embedded Software Engineering Kongress ESE-Kongress 2012*. 2012, pp. 683–689 (cit. on p. 134).

[Jer] J. Jeremiah. *Agile vs. waterfall: Survey shows agile is now the norm*. URL: `https://techbeacon.com/app-dev-testing/survey-agile-new-norm` (visited on 08/05/2019) (cit. on p. 74).

[JH97] John H. Bradley, John H., F. J. Herbert. 'The effect of personality type on team performance'. In: *Journal of Management Development* 16.5 (1997), pp. 337–353 (cit. on p. 125).

[JMT05] M. John, F. Maurer, B. Tessem. 'Human and social factors of software engineering'. In: *Proceedings of the 27th International Conference on Software Engineering*. 2005, p. 686 (cit. on p. 54).

[JPSG14] M. Jones, P. Palanque, A. Schmidt, T. Grossman, eds. *CHI 2014, one of a CHInd: Conference proceedings : Toronto, Canada, April 26 - May 1, 2014 ; the 32nd Annual ACM Conference on Human Factors in Computing Systems*. New York, NY: Assoc. for Computing Machinery, 2014 (cit. on p. 17).

[Jud12] K. H. Judy. 'Agile Values, Innovation and the Shortage of Women Software Developers'. In: *2012 45th Hawaii International Conference on System Sciences*. Ed. by R. H. Sprague. Piscataway, NJ: IEEE, 2012, pp. 5279–5288 (cit. on pp. 148–150).

[Jun21] C. G. Jung. *Psychologische Typen*. Vol. 6. Carl Gustav Jung, Gesammelte Werke. Zürich: Rascher, 1921 (cit. on p. 128).

[JY03] Jingqiu Shao, Yingxu Wang. 'A new measure of software complexity based on cognitive weights'. In: *Proceedings of the Canadian Conference on Electrical and Computer Engineering*. 2003, pp. 1333–1338 (cit. on p. 73).

[Kai85] K. M. Kaiser. 'The Relationship of Cognitive Style to the Derivation of Information Requirements'. In: *Newsletter ACM SIGCPR Computer Personnel* 10.2 (1985), pp. 2–12. URL: `http://delivery.acm.org/10.1145/1040000/1036375/p2-kaiser.pdf?ip=194.95.21.161&id=1036375&acc=ACTIVE%20SERVICE&key=2BA2C432AB83DA15%2EA95C2088598DDEA0%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1566808795_60284fe8dd40b051341a904aea6858bd` (visited on 08/26/2019) (cit. on p. 72).

[KBGW16] Z. Karimi, A. Baraani-Dastjerdi, N. Ghasem-Aghaee, S. Wagner. 'Links between the personalities, styles and performance in computer programming'. In: *Journal of Systems and Software* 111 (2016), pp. 228–241 (cit. on p. 19).

[KBJ14] A. Kramer, D. P. Bhave, T. D. Johnson. 'Personality and group performance: The importance of personality composition and work tasks'. In: *Personality and Individual Differences* 58 (2014), pp. 132–137 (cit. on p. 228).

[KC07] B. Kitchenham, S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007. (Visited on 05/02/2013) (cit. on pp. 53, 56, 159).

[KDG13] D. E. Knuth, E. G. Daylight, K. de Grave, eds. *The essential Knuth*. 1. ed. Vol. 3. Lonely Scholar conversations. Heverlee: Lonely Scholar, 2013 (cit. on p. 17).

[KMG14] T. Kanij, R. Merkel, J. Grundy. 'A Preliminary Survey of Factors Affecting Software Testers'. In: *2014 23rd Australian Software Engineering Conference (ASWEC): 7 - 10 April 2014, Sydney, New South Wales, Australia*. Piscataway, NJ: IEEE, 2014, pp. 180–189 (cit. on p. 209).

[KMG15] T. Kanij, R. Merkel, J. Grundy. 'An Empirical Investigation of Personality Traits of Software Testers'. In: *Eighth International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2015: May 18, 2015, Florence, Italy : proceedings*. Piscataway, NJ: IEEE, 2015, pp. 1–7. (Visited on 04/19/2018) (cit. on p. 142).

[Kri07] K. Krippendorff. 'Computing Krippendorff's alpha reliability'. In: *Departmental Papers (ASC)* (2007), p. 43 (cit. on p. 61).

[Kru15] J. K. Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. 2. ed. Amsterdam: AP Academic Press/Elsevier, 2015. URL: `http://www.contentreserve.com/TitleInfo.asp?ID=38F45CF6-6B5C-433C-85F8-A3568420927D&Format=50` (cit. on pp. 86, 87, 92–94).

[KW] Z. Karimi, S. Wagner. *The influence of personality on computer programming: a summary of a systematic literature review*. URL: http://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCIQFjAA&url=http%3A%2F%2Felib.uni-stuttgart.de%2Fopus%2Fvolltexte%2F2014%2F8987%2Fpdf%2FSLR_Summary.pdf&ei=8nEuVebmDsri08yxgbAD&usg=AFQjCNHSHTiWSNQZRuLiGWAH_OzUIqD4Ig&bvm=bv.90790515,d.ZWU (visited on 04/15/2015) (cit. on pp. 126, 127).

[KWB96] L. A. King, L. M. Walker, S. J. Broyles. 'Creativity and the Five-Factor Model'. In: *Journal of Research in Personality* 30.2 (1996), pp. 189–203 (cit. on p. 71).

[KWS09] M. Klawe, T. Whitney, C. Simard. 'Women in computing—take 2'. In: *Communications of the ACM* 52.2 (2009), pp. 68–76 (cit. on p. 148).

[LA04] K. Lee, M. C. Ashton. 'Psychometric Properties of the HEXACO Personality Inventory'. In: *Multivariate behavioral research* 39.2 (2004), pp. 329–358 (cit. on p. 31).

[LA09a] K. Lee, M. C. Ashton. *The HEXACO Personality Inventory - Revised*. 2009. URL: http://hexaco.org/hexaco-inventory (visited on 06/17/2019) (cit. on pp. 31, 87, 91, 160, 254).

[LA09b] K. Lee, M. C. Ashton. *The HEXACO Personality Inventory - Revised - Scaledescriptions*. 2009. URL: http://hexaco.org/scaledescriptions (visited on 01/10/2018) (cit. on pp. 128, 132–134).

[LAA09] L. Shoaib, A. Nadeem, A. Akbar. 'An empirical evaluation of the influence of human personality on exploratory software testing'. In: *2009 IEEE 13th International Multitopic Conference*. 2009, pp. 1–6 (cit. on p. 125).

[LBB+02] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, M. Zelkowitz. 'Empirical Findings in Agile Methods'. In: *Extreme programming and agile methods - XP/Agile Universe 2002*. Ed. by D. Wells. Vol. 2418. Lecture Notes in Computer Science. Berlin: Springer, 2002, pp. 197–207 (cit. on p. 150).

[LC10] J. Li, M. Chignell. 'Birds of a feather: How personality influences blog writing and reading'. In: *International Journal of Human-Computer Studies* 68.9 (2010), pp. 589–602 (cit. on p. 84).

[LH93] W. Li, S. Henry. ‘Object-oriented metrics that predict maintainability’. In: *Journal of Systems and Software* 23.2 (1993), pp. 111–122 (cit. on pp. 46, 47).

[LK77] J. R. Landis, G. G. Koch. ‘The Measurement of Observer Agreement for Categorical Data’. In: *Biometrics* 33.1 (1977), pp. 159–174 (cit. on p. 63).

[LL07] R. Lincke, W. Löwe. *Compendium of Software Quality Standards and Metrics - Version 1.0*. 2007. URL: `http://www.arisa.se/compendium/` (visited on 01/30/2019) (cit. on p. 48).

[LL13] J. Ludewig, H. Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. 3., korr. Aufl. Heidelberg: Dpunkt.verl., 2013 (cit. on pp. 33, 124).

[LLL] R. Lincke, J. Lundberg, W. Löwe. ‘Comparing software metrics tools’. In: *the 2008 international symposium*. Ed. by B. G. Ryder, A. Zeller, p. 131 (cit. on pp. 48, 51).

[LM06] M. Lanza, R. Marinescu. *Object-oriented metrics in practice: Using software metrics to characterize, evaluate, and improve the design of object-oriented systems ; with 8 tables*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2006. URL: `http://gbv.eblib.com/patron/FullRecord.aspx?p=302086` (cit. on pp. 46, 47, 116).

[LM14] S. A. Licorish, S. G. MacDonell. ‘Personality profiles of global software developers’. In: *EASE 2014: The 18th International Conference on Evaluation and Assessment in Software Engineering : London, May 12th-14th, 2014*. Ed. by M. Shepperd, T. Hall, I. Myrtveit. ICPS. New York, New York: Association for Computing Machinery, 2014, pp. 1–10 (cit. on pp. 19, 209).

[Mai12] C. Maier. ‘Personality Within Information Systems Research: A Literature Analysis’. In: *ECIS 2012 Proceedings*. 2012, Paper 101. URL: `http://aisel.aisnet.org/ecis2012/101` (cit. on p. 54).

[MHG+00] J. E. Mathieu, T. S. Heffner, G. F. Goodwin, E. Salas, J. A. Cannon-Bowers. ‘The influence of shared mental models on team process and performance’. In: *Journal of Applied Psychology* 85.2 (2000), pp. 273–283 (cit. on p. 124).

[Miz83] Mizuno. ‘Software Quality Improvement’. In: *Computer* 16.3 (1983), pp. 66–72 (cit. on p. 73).

[MKK06] S. C. Misra, V. Kumar, U. Kumar. ‘Success Factors of Agile Software Development’. In: *2006 International Conference on Software Engineering Research and Practise (SERP ’06)*. 2006, pp. 26–29. URL: `http://ww1.ucmss.com/books/LFS/CSREA2006/SER5088.pdf` (visited on 09/14/2012) (cit. on p. 149).

[MN05] E. Mannix, M. A. Neale. ‘What differences make a difference? The promise and reality of diverse teams in organizations’. In: *Psychological science in the public interest* 6.2 (2005), pp. 31–55 (cit. on p. 148).

[MNKR14] G. A. Macht, D. A. Nembhard, J. H. Kim, L. Rothrock. ‘Structural models of extraversion, communication, and team performance’. In: *International Journal of Industrial Ergonomics* 44.1 (2014), pp. 82–91. URL: `http://www.sciencedirect.com/science/article/pii/S0169814113001248` (cit. on p. 229).

[Moh18] D. Mohilo. *Top 10 der Programmiersprachen: Pythons Siegeszug in den aktuellen Rankings - JAXenter*. 2018. URL: `https://jaxenter.de/top-10-redmonk-pypl-tiobe-vergleich-python-73850` (visited on 07/15/2019) (cit. on p. 89).

[Moo91] J. E. Moore. ‘Personality characteristics of information systems professionals’. In: *Proceedings of the 1991 Conference on SIGCPR*. SIGCPR ’91. New York, NY and USA: ACM, 1991, pp. 140–155 (cit. on p. 203).

[MPP11] A. Marnewick, J.-H. Pretorius, L. Pretorius. ‘A perspective on human factors contributing to quality requirements: A cross-case analysis’. In: *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*. 2011, pp. 389–393 (cit. on p. 73).

[MRS+11] M. Mondal, M. S. Rahman, R. K. Saha, C. K. Roy, J. Krinke, K. A. Schneider. ‘An Empirical Study of the Impacts of Clones in Software Maintenance’. In: *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*. 2011, pp. 242–245 (cit. on p. 40).

[MSN10] O. Mazni, S.-L. Syed-Abdullah, Naimah Mohd Hussin. ‘Analyzing personality types to predict team performance’. In: *Proceedings of the 2010 International Conference on Science and Social Research*. 2010, pp. 624–628 (cit. on p. 227).

[MV14] M. A. Moore, J. F. Vucetic. ‘Identifying Effective IT Project Manager Personality Characteristics’. In: *GOCICT 2014: 2014 Annual Global Online Conference on Information and Computer Technology : 3-5 December 2014, Louisville, Kentucky : proceedings*. Ed. by E. Udoh. Piscataway, NJ: IEEE, 2014, pp. 96–100 (cit. on p. 210).

[MW05] L. M. Martin, L. T. Wright. ‘No gender in cyberspace? Empowering entrepreneurship and innovation in female-run ICT small firms’. In: *International Journal of Entrepreneurial Behaviour & Research* 11.2 (2005), pp. 162–178 (cit. on p. 149).

[Nei08] Neil McBride. ‘Using performance ethnography to explore the human aspects of software quality’. In: *IT & People* 21.1 (2008), pp. 91–111. URL: `https://doi.org/10.1108/09593840810860342,%20Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen` (cit. on p. 86).

[NHM00] NHMRC. *How to review the evidence: systematic identification and review of the scientific literature*. Canberra and Australia, 2000. (Visited on 07/16/2013) (cit. on p. 56).

[Non07] I. Nonaka. ‘The Knowledge-Creating Company’. In: *Harvard Business Review* July-August 2007 (2007), pp. 162–171. URL: `https://memberfiles.freewebs.com/84/90/65819084/documents/The%20Knowledge-Creating%20Company.pdf` (visited on 12/05/2020) (cit. on p. 148).

[NWC99] G. A. Neuman, S. H. Wagner, N. D. Christiansen. ‘The Relationship between Work-Team Personality Composition and the Job Performance of Teams’. In: *Group & Organization Management* 24.1 (1999), pp. 28–45 (cit. on p. 125).

[OS10] M. Omar, S.-L. Syed-Abdullah. ‘Identifying Effective Software Engineering (SE) Team Personality Types Composition using Rough Set Approach’. In: *Proceedings of the 2010 International Symposium in*

Information Technology. 2010, pp. 1499–1503 (cit. on pp. 19, 85, 228, 243).

[OW] J.-P. Ostberg, S. Wagner. ‘On Automatically Collectable Metrics for Software Maintainability Evaluation’. In: *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pp. 32–37 (cit. on p. 37).

[PBN+14] E. Papatheocharous, M. Belk, J. Nyfjord, P. Germanakos, G. Samaras. ‘Personalised continuous software engineering’. In: *1st International Workshop on Rapid Continuous Software Engineering : proceedings : June 3, 2014, Hyderabad, India*. Ed. by M. Tichy, J. Bosch, M. Goedicke, M. Larsson. New York, New York, USA: ACM Press, 2014, pp. 57–62 (cit. on p. 231).

[Per12] P. Perrotta. *A Short History of Software Engineering*. 2012. URL: `https://youtu.be/9IPn5Gk_0iM` (cit. on p. 17).

[PGF96] R. E. Park, W. B. Goethert, W. A. Florac. *Goal-Driven Software Measurement - A Guidebook*. Ed. by Software Engineering Institute. 1996. URL: `https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCYQFjAAahUKEwiXjbCuv9jHAhX`
`url=http%3A%2F%2Fwww.sei.cmu.edu%2Freports%2F96hb002.pdf&usg=AFQjCNGquDiPnmBRCO0uQjMm_kIiC7vpVQ&bvm=bv.101800829,d.bGQ` (visited on 09/02/2015) (cit. on p. 37).

[PHM10] W. E. Powell, D. S. Hunsinger, B. D. Medlin. ‘GENDER DIFFERENCES WITHIN THE OPEN SOURCE COMMUNITY: AN EXPLORATORY STUDY’. In: *Journal of Information Technology Management* XXI.4 (2010), pp. 29–37 (cit. on p. 149).

[Pir10] L. Pirzadeh. *Human Factors in Software Development: A Systematic Literature Review*. 2010 (cit. on p. 54).

[Pit05] D. J. Pittenger. ‘Cautionary comments regarding the Myers-Briggs Type Indicator’. In: *Consulting Psychology Journal: Practice and Research* 57.3 (2005), pp. 210–221. (Visited on 06/06/2018) (cit. on p. 27).

[Pit93] D. J. Pittenger. ‘The Utility of the Myers-Briggs Type Indicator’. In: *Review of Educational Research* 63.4 (1993), p. 467 (cit. on p. 125).

[PKS06] V. Pieterse, D. G. Kourie, I. P. Sonnekus. 'Software engineering team diversity and performance'. In: *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*. South African Institute for Computer Scientists and Information Technologists, 2006, pp. 180–186 (cit. on pp. 19, 85, 224).

[PP11] L. Pries-Heje, J. Pries-Heje. 'Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India'. In: *Agile Conference (AGILE), 2011*. Aug. 2011, pp. 20–28 (cit. on p. 151).

[PR08] M. Petticrew, H. Roberts. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008 (cit. on p. 55).

[Pre+99] L. Prechelt et al. 'Comparing Java vs. C/C++ Efficiency Differences to Interpersonal Differences'. In: *Commun. ACM* 42.10 (1999), pp. 109–112 (cit. on p. 71).

[Put17] B. Putano. *Most Popular and Influential Programming Languages of 2018*. 2017. URL: `https://stackify.com/popular-programming-languages-2018/` (visited on 11/29/2018) (cit. on p. 48).

[RB03] A. M. Ryan, M. R. Barrick. *Personality and work: Reconsidering the role of personality in organizations*. 1st ed. The organizational frontiers series. San Francisco, CA: Jossey-Bass, 2003. URL: `http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10300648` (cit. on pp. 124, 125).

[Rei11] D. J. Reifer. *Software Maintenance Success Recipes*. CRC Press, 2011 (cit. on pp. 38, 40).

[RH07] P. C. Rigby, A. E. Hassan. 'What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List'. In: *Proceedings of the Fourth International Workshop on Mining Software Repositories*. 2007, p. 23 (cit. on pp. 69, 72, 205, 215).

[RH08] P. Runeson, M. Höst. 'Guidelines for conducting and reporting case study research in software engineering'. In: *Empirical Software Engineering* 14.2 (2008), p. 131 (cit. on pp. 129, 132).

[Ric18] M. Richters. *Top 10 der Programmiersprachen: Die aktuellen Rankings im Vergleich - JAXenter*. 2018. URL: `https://jaxenter.de/redmonk-pypl-tiobe-rankings-vergleich-69108` (visited on 07/15/2019) (cit. on p. 89).

[RK03] J. Rilling, T. Klemola. 'Identifying comprehension bottlenecks using program slicing and cognitive complexity metrics'. In: *Proceedings of the 11th IEEE International Workshop on Program Comprehension*. 2003, pp. 115–124 (cit. on p. 73).

[RL16] M. Razavian, P. Lago. 'Feminine Expertise in Architecting Teams'. In: *IEEE Software* 33.4 (2016), pp. 64–71 (cit. on p. 146).

[RM15] N. Ramadan Darwish, N. M. Rizk. 'Multi-Dimensional Success Factors of Agile Software Development Projects'. In: *International Journal of Computer Applications* 118.15 (2015), pp. 23–30. URL: `https://www.researchgate.net/profile/Nancy_Rizk/publication/280773909_Multi-Dimensional_Success_Factors_of_Agile_Software_Development_Projects/links/55dec6a908ae79830bb591e5/Multi-Dimensional-Success-Factors-of-Agile-Software-Development-Projects.pdf` (visited on 12/06/2020) (cit. on p. 150).

[RMSA12] M. Rehman, A. K. Mahmood, R. Salleh, A. Amin. 'Mapping job requirements of software engineers to Big Five Personality Traits'. In: *Proceedings of the International Conference on Computer & Information Science*. 2012, pp. 1115–1122 (cit. on pp. 20, 68–70, 208).

[Rut01] R. H. Rutherfoord. 'Using Personality Inventories to Help Form Teams for Software Engineering Class Projects'. In: *SIGCSE Bull* 33.3 (2001), pp. 73–76. URL: `http://doi.acm.org/10.1145/507758.377486` (cit. on p. 125).

[SB04] H. Schaeper, K. Briedis. 'Kompetenzen von Hochschulabsolventinnen und Hochschulabsolventen, berufliche Anforderungen und Folgerungen für die Hochschulreform'. In: *HIS Kurzinformation Hochschul-Informations-System* 2004.A6 (2004), pp. 1–62 (cit. on p. 134).

[Sch97] K. Schwaber. 'SCRUM Development Process'. English. In: *Business Object Design and Implementation*. Ed. by J. Sutherland, C. Casanave, J. Miller, P. Patel, G. Hollowell. Springer London, 1997, pp. 117–134 (cit. on pp. 148, 150).

[SES11] J. Specht, B. Egloff, S. C. Schmukle. 'Stability and change of personality across the life course: The impact of age and major life events on mean-level and rank-order stability of the Big Five'. In: *Journal of Personality and Social Psychology* 101.4 (2011), pp. 862–882 (cit. on pp. 20, 26, 81).

[SG96] G. Saucier, L. R. Goldberg. 'Evidence for the Big Five in analyses of familiar English personality adjectives'. In: *European Journal of Personality* 10.1 (1996), pp. 61–77 (cit. on p. 126).

[Shn81] B. Shneiderman. 'Putting the human factor into systems development'. In: *Proceedings of the eighteenth annual computer personnel research conference*. ACM, 1981, pp. 1–13 (cit. on p. 54).

[SMGS09] N. Salleh, E. Mendes, J. Grundy, G. J. St. Burch. 'An empirical study of the effects of personality in pair programming using the five-factor model'. In: *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. 2009, pp. 214–225 (cit. on pp. 72, 241).

[SMGS10] N. Salleh, E. Mendes, J. Grundy, G. J. St. Burch. 'An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-factor Personality Model'. In: *Software Engineering, 2010 ACM*. Ed. by J. Kramer, J. Bishop, P. Devanbu, S. Uchitel. New York, New York, USA: ACM Press, 2010, pp. 577–586. URL: http://doi.acm.org/10.1145/1806799.1806883 (cit. on pp. 72, 245).

[Smi89] D. C. Smith. 'The personality of the systems analyst: an investigation'. In: *SIGCPR Comput. Pers.* 12.2 (1989), pp. 12–14 (cit. on p. 212).

[SNA09] L. Shoaib, A. Nadeem, A. Akbar. 'An empirical evaluation of the influence of human personality on exploratory software testing'. In: *Proceedings of the IEEE 13th International Multitopic Conference*. 2009, pp. 1–6 (cit. on pp. 69, 72, 206, 241).

[SPD+05] G. Succi, W. Pedrycz, S. Djokic, P. Zuliani, B. Russo. 'An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite'. In: *Empirical Software Engineering* 10.1 (2005), pp. 81–104 (cit. on p. 37).

[SPWK07] S.-T. Shen, S. D. Prior, A. S. White, M. Karamanoglu. 'Using personality type differences to form engineering design teams'. In: *Engineering Education* 2.2 (2007), pp. 54–66 (cit. on p. 125).

[SR05] Susan Bergin, Ronan Reilly. 'Programming: factors that influence success'. In: *SIGCSE Bull* 37.1 (2005), pp. 411–415 (cit. on p. 73).

[SS14] A. B. Soomro, N. Salleh. 'A systematic review of the effects of team climate on software team productivity'. In: *2014 Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*. Piscataway, NJ: IEEE, 2014, pp. 1–7 (cit. on p. 74).

[SSAO04] I. Samoladas, I. Stamelos, L. Angelis, A. Oikonomou. 'Open source software development should strive for even greater code maintainability'. In: *Communications of the ACM* 47.10 (2004), pp. 83–87 (cit. on p. 33).

[TB03] R. P. Tett, D. D. Burnett. 'A personality trait-based interactionist model of job performance'. In: *Journal of Applied Psychology* 88.3 (2003), pp. 500–517 (cit. on p. 125).

[TB18] The Myers, Briggs Foundation. *The Myers & Briggs Foundation - MBTI® Basics*. 2018. URL: `https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/home.htm?bhcp=1` (visited on 08/20/2018) (cit. on pp. 125, 127).

[Tur] M. Turner. *Digital Libraries and Search Engines for Software Engineering Research: An Overview*. URL: `https://community.dur.ac.uk/ebse/resources/notes/tools/SearchEngineIndex_v5.pdf` (cit. on p. 56).

[Vas14] B. Vasilescu. 'Human aspects, gamification, and social media in collaborative software engineering'. In: *36th International Conference on Software Engineering (ICSE Companion 2014) : proceedings : May 31 - June 7, 2014, Hyderabad, India*. Ed. by P. Jalote, L. Briand, A. van der Hoek. New York, New York, USA: ACM Press, 2014, pp. 646–649 (cit. on p. 86).

[VCPR12] D. Varona, L. F. Capretz, Y. Pinero, A. Raza. ‘Evolution of software engineers’ personality profile’. In: *SIGSOFT Softw. Eng. Notes* 37.1 (2012), pp. 1–5 (cit. on pp. 19, 70, 215).

[VDS09] V. Vinod, J. Dhanalakshmi, S. Sahadev. ‘Software Team Skills on Software Product Quality’. In: *Asian Journal of Information Technology* 8.1 (2009), pp. 8–13 (cit. on p. 225).

[Ver18] Verbi GmbH. *Ähnlichkeitsanalyse für Dokumente - MAXQDA - The Art of Data Analysis*. 2018. URL: `https://www.maxqda.de/hilfe-max18/mixed-methods/aehnlichkeitsanalyse-fuer-dokumente` (visited on 08/22/2018) (cit. on p. 131).

[Vik] Viking Code School. *A Brief History of Software Engineering*. URL: `https://www.vikingcodeschool.com/software-engineering-basics/a-brief-history-of-software-engineering` (visited on 04/08/2019) (cit. on pp. 17, 18).

[VLC14] D. Varona, Y. Lizama-Mué, L. F. Capretz. ‘A comparison of junior and senior software engineering students’ personalities’. In: *7th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2014): Proceedings : June 2-June 3, 2014, Hyderabad, India*. Ed. by H. Sharp, R. Prikladnicki, A. Begel, de Souza, Cleidson R. B. New York, NY: Association for Computing Machinery, Inc, 2014, pp. 131–132 (cit. on p. 217).

[Vog12] Vogel Communications Group GmbH & Co. KG. *Software-Wartbarkeit – der unterschätzte Wettbewerbsvorteil*. 2012. URL: `https://www.elektronikpraxis.vogel.de/software-wartbarkeit-der-unterschaetzte-wettbewerbsvorteil-a-386818/` (visited on 10/29/2018) (cit. on p. 46).

[VP14] M.-F. Vaida, P. G. Pop. ‘Grouping strategy using Enneagram typologies’. In: *2014 IEEE International Conference on Automation, Quality and Testing, Robotics: 22 - 24 May 2014, Cluj-Napoca, Romania*. Ed. by L. Miclea. Piscataway, NJ: IEEE, 2014, pp. 1–6 (cit. on p. 233).

[Wag13] S. Wagner. *Software Product Quality Control*. Berlin, Heidelberg and s.l.: Springer Berlin Heidelberg, 2013. URL: `http://dx.doi.org/10.1007/978-3-642-38571-1` (cit. on p. 46).

[WB07] E. Whitworth, R. Biddle. 'The Social Nature of Agile Teams'. In: *Agile Conference (AGILE), 2007*. Aug. 2007, pp. 26–36 (cit. on p. 150).

[WB15] E. Weilemann, P. Brune. 'Less Distress with a Scrum Mistress?' In: *Proceedings of the 24th Australasian Software Engineering Conference (ASWEC 2015): Adelaide, Australia, 28 September - 1 October 2015*. Ed. by F.-C. Kuo. New York, NY, USA: ACM, 2015, pp. 3–7 (cit. on pp. 23, 146).

[WC17] D. Winsborough, T. Chamorro-Premuzic. *Great Teams Are About Personalities, Not Just Skills*. 2017. URL: `https://hbr.org/2017/01/great-teams-are-about-personalities-not-just-skills` (visited on 03/21/2018) (cit. on p. 124).

[WDW] S. Wagner, F. Deissenboeck, S. Winter. 'Managing quality requirements using activity-based quality models'. In: *the 6th international workshop*. Ed. by B. Wong, p. 29 (cit. on pp. 35, 40).

[WGC+08] A. W. Woolley, M. E. Gerbasi, C. F. Chabris, S. M. Kosslyn, J. R. Hackman. 'Bringing in the Experts'. In: *Small Group Research* 39.3 (2008), pp. 352–371 (cit. on p. 124).

[WGW19] M. Wyrich, D. Graziotin, S. Wagner. 'A theory on individual characteristics of successful coding challenge solvers'. In: *PeerJ Computer Science* 5.2 (2019), e173 (cit. on p. 126).

[WH09] T. Walle, J. E. Hannay. 'Personality and the nature of collaboration in pair programming'. In: *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. 2009, pp. 203–213 (cit. on p. 226).

[WHJ+07] A. W. Woolley, J. R. Hackman, T. E. Jerde, C. F. Chabris, S. L. Bennett, S. M. Kosslyn. 'Using brain-based measures to compose teams: how individual capabilities and team collaboration strategies jointly shape performance'. In: *Social neuroscience* 2.2 (2007), pp. 96–105 (cit. on p. 124).

[Wir] N. Wirth. *A Brief History of Software Engineering*. URL: `https://inf.ethz.ch/personal/wirth/Miscellaneous/IEEE-Annals.pdf` (visited on 04/08/2019) (cit. on p. 17).

[WK14] M. Wiesche, H. Krcmar. ‘The relationship of personality models and development tasks in software engineering’. In: *Proceedings of the 2014 Conference on Computers and People Research, May 29 - 31, 2014, Singapore, Singapore*. Ed. by D. Joseph. New York, NY: ACM, 2014, pp. 149–161 (cit. on p. 75).

[Woo79a] C. K. Woodruff. ‘Consideration of selected personality-job satisfaction constructs relevant to project management in data processing organizations’. In: *Proceedings of the sixteenth annual SIGCPR conference*. 1979, pp. 13–19 (cit. on p. 212).

[Woo79b] C. K. Woodruff. ‘Personality profiles of male and female data processing personnel’. In: *SIGCPR Comput. Pers.* 8.1 (1979), pp. 10–14 (cit. on pp. 18, 211).

[WW98] J. L. Wynekoop, D. B. Walz. ‘Revisiting the perennial question: are IS people different?’ In: *SIGMIS Database* 29.2 (1998), pp. 62–72 (cit. on p. 213).

[www18] www.tiobe.com. *TIOBE Index | TIOBE - The Software Quality Company*. 2018. URL: `https://www.tiobe.com/tiobe-index/` (visited on 11/29/2018) (cit. on p. 48).

[YEMT05] S. M. Young, H. M. Edwards, S. McDonald, J. B. Thompson. ‘Personality characteristics in an XP team: a repertory grid study’. In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), pp. 1–7 (cit. on pp. 67–69, 204).

[YOCC17] M. Yilmaz, R. V. OConnor, R. Colomo-Palacios, P. Clarke. ‘An Examination of Personality Traits and How They Impact on Software Development Teams’. In: *Information and Software Technology* 86.C (2017), pp. 101–122. URL: `https://doi.org/10.1016/j.infsof.2017.01.005` (cit. on pp. 19, 159).

[YR08] Z. Yanyan, X. Renzuo. ‘The Basic Research of Human Factor Analysis Based on Knowledge in Software Engineering’. In: *Proceedings of the International Conference on Computer Science and Software Engineering*. 2008, pp. 1302–1305 (cit. on p. 73).

[YT04] H.-L. Yang, J.-H. Tang. 'Team structure and team performance in IS development: a social network perspective'. In: *Information & Management* 41.3 (2004), pp. 335–349. URL: http://www.sciencedirect.com/science/article/pii/S0378720603000788 (cit. on p. 220).

[ZOY14] F. A. Zaraket, M. Olleik, A. A. Yassine. 'Skill-based framework for optimal software project selection and resource allocation'. In: *European Journal of Operational Research* 234.1 (2014), pp. 308–318. URL: http://www.sciencedirect.com/science/article/pii/S037722171300790X (cit. on p. 127).

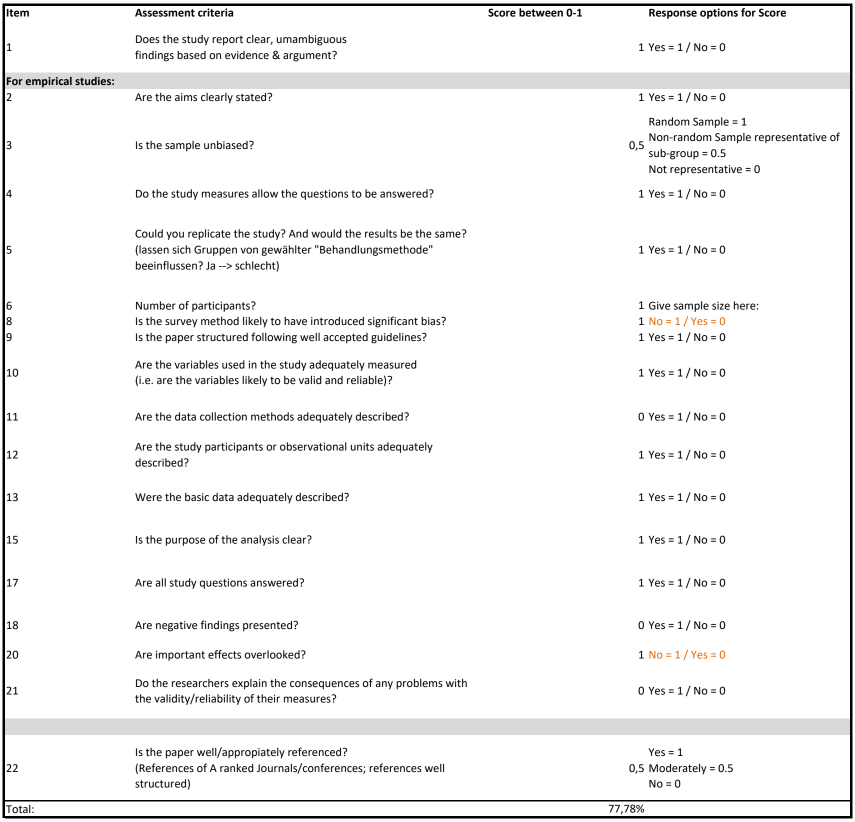
List of Figures

List of Tables

List of Definitions

Appendix A

Appendix A



Item	Assessment criteria	Score between 0-1	Response options for Score
1	Does the study report clear, umambiguous findings based on evidence & argument?	1	Yes = 1 / No = 0
For empirical studies:			
2	Are the aims clearly stated?	1	Yes = 1 / No = 0
3	Is the sample unbiased?	0,5	Random Sample = 1 Non-random Sample representative of sub-group = 0.5 Not representative = 0
4	Do the study measures allow the questions to be answered?	1	Yes = 1 / No = 0
5	Could you replicate the study? And would the results be the same? (lassen sich Gruppen von gewählter "Behandlungsmethode" beeinflussen? Ja --> schlecht)	1	Yes = 1 / No = 0
6	Number of participants?	1	Give sample size here:
8	Is the survey method likely to have introduced significant bias?	1	No = 1 / Yes = 0
9	Is the paper structured following well accepted guidelines?	1	Yes = 1 / No = 0
10	Are the variables used in the study adequately measured (i.e. are the variables likely to be valid and reliable)?	1	Yes = 1 / No = 0
11	Are the data collection methods adequately described?	0	Yes = 1 / No = 0
12	Are the study participants or observational units adequately described?	1	Yes = 1 / No = 0
13	Were the basic data adequately described?	1	Yes = 1 / No = 0
15	Is the purpose of the analysis clear?	1	Yes = 1 / No = 0
17	Are all study questions answered?	1	Yes = 1 / No = 0
18	Are negative findings presented?	0	Yes = 1 / No = 0
20	Are important effects overlooked?	1	No = 1 / Yes = 0
21	Do the researchers explain the consequences of any problems with the validity/reliability of their measures?	0	Yes = 1 / No = 0

22	Is the paper well/appropiately referenced?
	(References of A ranked Journals/conferences; references well structured)
		0,5	Yes = 1
			Moderately = 0.5
			No = 0

Total:		77,78%	

Figure A.1.: Example of a filled out evaluation sheet we used for the quality assurance of the studies.

Table A.1.: Studies concerning the matching of personality traits to distinct software engineering roles. ⚠
The following contents are citations. Due to readability reasons the "" and [...] were ommitted, also some sentences were rearranged. The sources are given in the very left column.

Paper	**Year**	**Personality Type Test**	**Research Question(s)**	**Key Findings**
[BK81]	1981	MBTI	Are users (not end users) and systems staff significantly different on personality characteristics, that relate to problem solving?	No differences between users and systems personnel on the EI, TF, JP dimensions; significant differences on the SN dimension
[Moo91]	1991	16 Personality Factor Questionnaire	a: Do different groups of information systems personnel differ significantly in personality characteristics? b: Which personality factors best define the differences?	Application programmers (P) and data processing managers (M) tended to be more experimenting, imaginative, forthright (less astute) than application systems analysts (A) and technical programmers (T); M tended to be more lax and impulsive (less controlled and socially precise), more assertive, more abstract-thinking than P
[Bis95]	1995	MBTI	How are cognitive style and personality variables related to programming?	Problem representation: Introversion/Extraversion, coding: Thinking/Feeling

[CT98]	1998	MBTI	Which personality attributes of computing professionals are best suited to system analysis, system design and programming?	system analysis: combination of Extraversion and Intuition, system design: Intuition and Thinking, Sensing and Judging, Introvert and Sensing, and Thinking and Perceiving, programming: Sensing with attention to detail, Intuition and Thinking
[YEMT05]	2005	-	Which personality characteristics matter most within the systems development roles?	team leader: analytical, take responsibility for a team, ability to define appropriate plans for achieving goals whilst also having the support of and support for the team, provide ideas (at both, the large and the small scale), carry these through in association with the team; technical lead: analytical, capable of taking a project forward, teamoriented, prepared to take advice to seek out the best route forward; technical architect: same as technical lead, only difference being the extreme focus on each

cont.				characteristic; operations manager: rigorous and disciplined, capable of leading a project with relevant interpersonal skills; good team member: analytical with good interpersonal skills and passion for extending his knowledge base and passing this on to others; bad team member: domineering management with little inclination to use or share knowledge and offer support to other team members
[DG07]	2007	MBTI	What is the link between personality type and code-review ability?	NT individuals were better at code-review tasks than the non-NT people (EI, SN, TF, JP)
[RH07]	2007	Big Five	What is the personality type of OSS developers?	The personalities are different from the baseline and other developers on the traits of Extroversion and Openness
[BT]	2008	Big Five	Which personality traits are good predictors for project manager (PM) success?	Conscientiousness and Openness are positive predictors of project manager (PM) success

[Ber08]	2008	-	What is the “real” role of a software architect in large projects?	An architect must have profound theoretical knowledge and ability to listen, to motivate and coach, attention to detail, a healthy dose of scepticism, up and down communication, decision making, open mind, visionary
[SNA09]	2009	MBTI	Which personality traits influence exploratory testing?	People having extrovert personality types are good exploratory testers.
[BH]	2010	NEO-FFI	Which impact do personality traits of IS managers have on the relative importance they ascribe to evaluation criteria in ERP selection?	Neurotic IS managers put emphasis on functional aspects and reliability; Conscientious IS managers are more likely to consider cost as crucial evaluation factor in ERP selection; Agreeable IS managers emphasize the ease of use and vendor support; Open IS managers focus more on the ease of customization; Extravert IS managers attach importance to vendor support and ease of implementation

[CA10b]	2010	MBTI	Discussion of the connections between human factors and the process of developing software	System analyst: EF, software designer: NT, programmer: IST, testing: SJ, maintenance: SP
[DSZY11]	2011	-	Which dimensions of cognitive style do software developers prefer?	Compared to the common testees, software developers'verbal/imagery dimenstion of the cognitive style belongs to the significant imagery preference. As to wholist/analytic dimension, software developers are wholist style but non-significant.
[CdSM12]	2012	-	Fixing of concepts from knowledge management and creativity in relation with new software engineering trends (agile methods)	client - detective, client and manager - explorer, developer - artist, tester and tracker - engineer, tracker and client - judge, client in his organization - producer (Mapping roles in XP to roles in a creative team)

[RMSA12]	2012	Big Five	Which personality traits fit best to roles in a software development process?	Software/system analyst: Extraversion, Agreeableness, software designer: Agreeableness, Openness to Experience, software programmer: Openness to Experience, Extraversion, Agreeableness, software tester: Openness to Experience, Conscientiousness, software maintenance: Conscientiousness, Openness to Experience
[HT14]	2014	-	evaluate personal and interpersonal skills of project managers	Empathy: It is vital to address issues before they turn into a full blown conflict. Conflict Resolution: Learning to resolve conflicts will lead to better team playing. Negotiation skills: Trying to find win-win situations across multiple stakeholders with varied expectations. Communication skills: Meeting information management needs of stakeholders.

[KMG14]	2014	questions on general abilities and characteristics; Big Five like	the opinion of software testing practitioners on the factors influencing their effectiveness	intelligence, dedication, interpersonal skills, and motivation were viewed as crucial in being an effective software tester
[LM14]	2014	Big Five	the influence of global software practitioners' personality profiles from a psycholinguistic perspective	Top Members demonstrated more openness to experience than the Other practitioners. Practitioners involved in usability-related tasks were found to be highly extroverted, coders were most neurotic and conscientious. High levels of organizational and inter-personal skills may be useful for those operating in distributed settings, and personality diversity is likely to boost team performance.

[MV14]	2014	Big Five	IT project manager: relationship of each of the Big-Five personality traits to the project success factors of on time, desired quality achieved, and within budget	IT project manager: agreeableness is strongly significant in achieving project outcomes based on time; Neuroticism had a strongly negative correlation with time. Agreeableness had a strongly positive correlation with time, conscientiousness was the only personality trait that was correlated with all three-success components of time, quality, and budget

Table A.2.: Studies concerning the distribution of personality types among software engineers and in comparison to the population.⚠ The following contents are citations. Due to readability reasons the "" and [...] were ommitted, also some sentences were rearranged. The sources are given in the very left column.

Paper	**Year**	**Personality Type Test**	**Research Question(s)**	**Key Findings**
[Woo79b]	1979	Murray‘s variables of personality	Are there differences in the distribution of personality types between data processing personnel and general population? Are there differences between the distribution of data processing personnel between males and females?	Data processing personnel possesses higher needs for achievement, cognitive structure, endurance, harmavoidance, order; lower needs for aggression, change, exhibition, impulsivity, play, social recognition; data processing males: significantly higher needs for endurance, harmavoidance than general population counterparts, also higher needs for achievement, cognitive structure, autonomy, dominance, harmavoidance, order, understanding; lower need for affiliation than general population counterparts

[Woo79a]	1979	Murrey‘s variables of personality	Comparison of personality traits between general population and system analysts, programmer analysts and programmers	personality traits of people in software industry are different than the general population; data processing functional groups possess a need for achievement greater than the general population, the level of nurturance is a little bit higher than males of the general population
[Smi89]	1989	MBTI	What is the distribution of personality types of system analysts?	The majority of the sample analysed in insurance company have a strong technical oriantation and a very conservative outlook. The most frequent types are ISTJ (35%) and ESTJ (30%), sensing (81%), thinking (89%), judging (86%); the most frequent combination of two preferences is TJ (76%).

[WW98]	1998	ACL (California Psychological Inventory Adjective Check List)	RQ1: do personality characteristics of IS employees differ from those of the general population? RQ2: Are the personality characteristics of programmers more different from those of the general population than the scores of developers in less traditional roles (systems analysts and managers)? RQ3: Are the personality characteristics for programmers different from those of developers in other roles (systems analysts and managers)?	RQ1: IS professionals differed from adult population norms on 19 of the 24 scales examined; IS professionals exceeded population norms on scales relating to ambition, logic, conservatism; falling below norms on scales relating to submissive behavior; RQ2: managers and systems analysts differ from the general population on more scales than do programmers; programmers are more conventional, conscientious, logical and analytical than the general population, less social, gregarious and imaginative,

cont.				scored lower than population norms on the Aggression, change, exhibition scales, higher on self-control scale; analysts and managers scored significantly lower than the general population on abasement scale, higher than the population norms on the achievement, dominance, endurance, creative personality, ideal self, military leadership, personal adjustment, self-confidence scales, managers scored lower on the High Origence-Low Intellectence and Abasement scales; analysts scored higher on the Affiliation scale; RQ3: differences were statistically significant for achievement, dominance, exhibition, self-conficence
[Cap03]	2002	MBTI	What is the distribution of MBTI types among software engineers?	Largest single type among subjects: ISTJ, second most type: ESTJ, (ISTJ+ISTP+ESTP+ESTJ)>50 %

[RH07]	2007	Big Five	What is the personality type of OSS developers?	Two developers who were responsible for two major Apache releases had similar personalities. Their personalities were different from the baseline and other developers on the traits of extroversion and openness.
[VCPR12]	2012	MBTI	What is the distribution of personality types among software engineers and what are the changes of distribution chronologically?	The distribution of MBTI personality types among software engineers changed over the last 30 years. INTJ+ISTJ are decreasing, ESTP is increasing; ESTP is still lower than INTJ and ISTJ; Extroverts, Judging and Feelers are increasing, Introverts, Perceiving and Thinkers are decreasing; Thinkers outnumber Feelers.

[FL14]	2014	MBTI; Belbin Team Roles	What is the personality type pattern that is often associated with business analysts and developers in the software development lifecycle? Is there a direct correlation between the individual's performance rating and his/her personality?	Business analysts differ from developers in the sense that most of the aforementioned have revealed to be more extroverts than introverts. ... the majority of both business analysts and developers leaned more towards the sensing than the intuition character trait. ... all the business analyst candidates and developer candidates are more ’thinking’ than ’feeling’ in their personality types when it comes to ways of making decisions. ... both these roles in the IT world are more judging than perceptive in nature ... strongest Belbin role depicted for a business analyst position would be that of a resource investigator ... The complete finisher is the role that came out as the strongest within the developer data set.

[VLC14]	2014	MBTI	MBTI personality distribution of first (junior) and fifth (senior) year software engineering students at the University of Informatics Sciences, in Havana (Cuba), aiming to infer attrition rates among software engineering students	ESTPs and ENTPs are overrepresented in both samples, whereas INTJs and INTPs are underrepresented.

Table A.3.: Search results which deal with personality type and team performance.⚠ The following contents are citations. Due to readability reasons the "" and [...] were ommitted, also some sentences were rearranged. The sources are given in the very left column.

Paper	Year	Personality Type Test	Research Question(s)	Definition of Performance	Key Findings
[BPW02]	2002	Big Five	What is the influence of extraversion, variation in extraversion, virtual group interaction style on contextual performance? What is the influence of extraversion, variation in extraversion on virtual group interaction style? What is the influence of group	-	Expertise is the most powerful predictor of team performance (quality of the team solution is related to the amount of expertise available in the team). The importance of extraversion and variation in extraversion is limited in the models. The researchers found no role for team size.

cont.	expertise, virtual group interaction style, best members on task performance? What is the influence of group expertise on virtual group interaction style? What is the influence of best member on group expertise?	-	Expertise is the most powerful predictor of team performance (quality of the team solution is related to the amount of expertise available in the team). The importance of extraversion and variation in extraversion is limited in the models. The researchers found no role for team size.

[YT04]	2003	-	(1) explore the variations in team structure during different phases of projects; (2) identify the relationship between team structure and information system development (ISD) team performance; and (3) explore the relationship of a social relational map (sociogram) and ISD performance	performance = grade as a mean of grades for analysis, design and implementation	cohesion index fluctuated between different information system development phases; group cohesion is positively related to overall performance; group conflict indexes not significantly correlated with overall performance; group characteristics (cohesion, conflict) fluctuated in different phases, in larger stages much less cohesion occurred and the advice network seemed to be very important; group structures seemed to be a critical factor for good performance

[BPW04]	2004	Big Five	influence of extraversion, variation in extraversion, virtual group interaction style on **contextual performance**; influence of extraversion, variation in extraversion on **virtual group interaction style**; influence of group expertise, virtual group interaction style, best members	task measure of performance: 1) team error: absolute difference between the team's consensus solution and the expert solution, 2) team synergy: subtracting the team (consensus) performance measure from the best member's performance	extraversion leads to constructive and/or aggressive styles, differences in extraversion within a team lead to passive styles, it is mostly group styles that have predictive power on outcomes; expertise decreases team errors and promotes synergy, it is more difficult for the team to outperform its most expert member, a high level of expertise leads to

cont.	on **task performance**; influence of group expertise on **virtual group interaction style**; influence of best member on **group expertise**	task measure of performance: 1) team error: absolute difference between the team's consensus solution and the expert solution, 2) team synergy: subtracting the team (consensus) performance measure from the best member's performance	non-constructive behaviors; extraversion increases team errors; there has to be adequate knowledge within the team and a willingness to share and build upon that knowledge base toward synergistic solutions that are superior to those of the best individual within a virtual team, in virtual settings extraversion is an important personality trait to promote that interaction, teams with lower variances in extraversion do best - especially in teams with good knowledge to start off with

[GL04]	2004	MBTI	What is the relationship between personality traits of team members and team performance in small IS teams?	self evaluation of team performance and quality of work	performance of the team is better, if: - team leader is N(Intuitive) and F(eeling), - system analyst is T(hinking), - programmer is E(xtroverted)
[BZG05]	2005	-	(1) What is the distribution of informal roles in student software engineering teams? (2) What are the results of a comparison of (1) to the concept of functional group roles?	-	The typical group roles the members of the software engineering teams assumed basically match the concept of functional group roles as defined by Benne et al [BS48]. They found the existence of four distinct functional group roles in sw engineering teams: 1) group task roles, 2) group building roles, 3) "typical programmers", 4) individual roles.

cont.					Gender aspects have a significant impact on the distribution of roles.
[PKS06]	2006	Keirsey-Bates Temperament Sorter	What is the correlation between personality diversity and success of teams? What is the correlation between competence and team success?	team ability: grades achieved for theoretical tests in the course as measure for individual ability (three tests), ability of the team = average individual evaluation mark of the team's members	Personality diversity of the teams correlated strongly with team success during inception phase. This correlation weakened during the course of the year while correlation between competence and success started slightly weaker than personality diversity during inception phase, but grew very strong towards the completion of the team projects.

[VDS09]	2009	-	How can the relationship between the skill of the software development team and several measures of software product quality be quantified? What is the collective skill of a development team?	software development team skills: product development skills (technical skills on language, programming, application and hardware experience) and business development skills (project management skills like motivation, leadership, communication coordination); skill ratings of individual team members	development team skill has a significant effect on the quality of a software product; higher proportions of skilled and experienced staff in the technical, people management and implementation life cycle phases increased the adequacy of the people management modules and source code units and had a similar albeit more muted effect on the functional implementation completeness and correctness and on the volatility of the software specification.

[WH09]	2009	Big Five	What is the nature of pair programming collaboration? What are the postulated effects of personality on pair programming collaboration?	-	Personality generally affects the type of collaboration that occurs in pairs. Different levels of a given personality trait between two pair members increases the amount of communication-intensive collaboration exhibited by a pair.

[MSN10]	2010	MBTI, Keirsey-Bates Temperament Sorter	Which personality types can affect team performance?	team performance was measured by the grades	Heterogeneous teams were more creative when developing high quality software, less effective when the project were less challenging because members' abilities were not adequately being addressed for solving software projects. Teams with homogeneous members were more comfortable with less challenging projects because members' skills and abilities were of similar level, thus hindering them from developing high quality software.

[OS10]	2010	MBTI	What are effective personality type compositions in software engineering teams?	grades of the project	A balance of the personality types Sensing, Intuitive, Thinking and Feeling assisted teams in achieving higher software quality. Extroverted members also had an impact on team performance.
[KBJ14]	2014	Big Five	Does group members' Big Five personality composition affect the group's performance?	conjunctive task: total number of blocks the group contributed over ten trials; additive task: total number of connected puzzle pieces completed by the group	Variability in extraversion is positively related to group performance on the additive (where group performance is based on the sum of efforts of all group members; here jigsaw puzzle) task but not on the conjunctive task (where group performance is based on the performance of the weakest group member; here tower building task).

cont.					Conversely, neuroticism maximum score is negatively related to group performance on the conjunctive task but not on the additive task.
[MNKR14]	2013	Big Five	RQ1: Does team communication mediate the relationship between team extraversion and overall team performance? RQ2: Does by-role communication mediate the relationship between team extraversion and overall team performance?	team performance = (correct identification responses)/(total number of time windows assigned for the unknown identification task)	no direct relationship between team extraversion and performance; communication mediates the relationship between the extraversion of the team's members and the overall team performance (depending on individual and team communication measures); communication

cont.	RQ3: Does by-role communication mediate the relationship between team extraversion and by-role aggregation of team performance?	mediates the relationship between team extraversion and overall team performance (affected by the specific roles within the dyad); by-role communication mediates the relationship between team extraversion and by-role aggregation of team performance, with aggregated team performance; each communication measure (number of words, utterances, durations) plays a part in mediating the personality-performance relationship

[PBN+14]	2014	Big Five / FFM NEO-IPIP test	A personalised approach of integrating human factors in continuous software engineering including personality factors and cognitive processing characteristics	subjective satisfaction of team members	the individuals in the highest performing pairs have had a correlation in their personalities. They also matched most of the times in terms of the Neuroticism (N) characteristic, being in quite low value. This means that there is an indication that individuals with correlated personality types might perform better as a pair than

cont.	non-correlated personalities. Individualised continuous software engineering can potentially serve efficiently the individual needs and preferences of the teams and the people involved in the development process.

[VP14]	2014	Enneagram Typologies (RHETI test), MBTI	Dividing a class of students into groups and assigning roles, based on an appropriate grouping strategy, in order to maximize the efficiency for team-based projects; grouping is done taking into account student typologies (Enneagram) and their NLP profile	team performance is subjective measure of the authors	groups created using this method perform better than groups that don't take into account the personality factor, in terms of selfdevelopment and group communication

Table A.4.: Search results which deal with personality type and software quality.⚠ The following contents are citations. Due to readability reasons the "" and [...] were ommitted, also some sentences were rearranged. The sources are given in the very left column.

Paper	Year	Personality Type Test	Study Subjects	Research Question(s)	Software Quality Attribute and/or Measure	Key Findings
[CWW03]	2003	Adjective Check List, Big Five / Five Factor Model	Students and practitioners/-experts	Comparison of personality characteristics of exceptional, experienced application software developers with the personality characteristics of junior and senior level IS and CS students	grade point average (for students), job performance (for software developers)	- exceptional students (IS+CS): Conscientious-ness+Extraversion; - IT-developers: Extraversion + Conscientiousness

[DM05]	2005	Big Five	Students	What is the relationship between individual characteristics and programmer performance?	specification conformance, structural complexity (LCOM, DIT, CBO, NOC, RFC, WMC, total lines of code, total lines of comments, total blank lines)	It is not clear, whether the personality profile of the participants represent the ideal profile of an exceptional programmer. Results indicate that programmer selection using personality profiles may be problematic because little

cont.	research done to date has identified the personality profile for an excellent programmer. GPA and age had a major impact on individual task performance. Personality is distal from programming performance and the effect is mediated by intelligence and academic achievement.

[CH06]	2006	Rosenberg's Self-Esteem Scale, Judge's Generalized Self-Efficacy Scale, Levenson's Locus of Control Scale, Eysenck and Eysenck's Neuroticism scale	Students	RQ1: Is theoretical value belief a valid predictor of proficiency in OO programming task performance? RQ2: Is cognitive ability a valid predictor of proficiency ability, and personality together valid predictors of OO programming task performance?	programming performance: grades (3 written exams, 2 timed laboratory exams, comprehensive programming assignment (from problem formulation	RQ1-3: individually, each of the constructs of theoretical value belief, cognitive ability, and personality exhibits a predictive relationship with OO programming performance; theoretical value belief and personality are both extremely

cont.				in OO programming task performance? RQ3: Is personality a valid predictor of OO programming task performance? RQ4: Are theoretical value belief, cognitive	to implementation))	potent predictors of performance; a person with a strong theoretical value belief is equally as likely to succeed in programming performance as a person with a strong personality (high self-esteem, high self-efficacy, high locus of control, low neuroticism)
[DG07]	2007	MBTI	Students	What is the link between personality type and code-review ability?	Code review ability measured by task score	NT individuals were better at code review task than the non-NT people (EI, SN, TF, JP)

[AGJ09]	2008	Big Five	Students	What are the relationships between personality, team processes, task characteristics, product quality and satisfaction in software development teams?	decomposition and modularization (number of modules and coupling), testability (number of defects detected by the automatically executed test case set),	the teams with the highest job satisfaction are precisely the ones whose members score highest for the personality factors agreeableness and conscientiousness. The satisfaction levels are also higher when the members can

cont.	functionality (number of satisfied requirements), reusability (number of reused modules), programming style (guidelines defined at the subject website), team member participation (checklist of laboratory session observations)	decide how to develop and organize their work. On the other hand, the level of satisfaction and cohesion drops the more conflict there is between the team members. Finally, the teams exhibit a significant positive correlation between the personality factor extraversion and software product quality.

[SMGS09]	2009	Big Five	Students	Do differences in personality traits affect the academic performance of students who pair programmed?	pair programming effectiveness was measured by academic performance and students' satisfaction	The study did not provide any evidence for distinguishing the performance of paired students between personality groups. Positive correlations between conscientiousness and assignments' scores. The students' test performance was found to be positively correlated with openness to experience.
[SNA09]	2009	MBTI	Students	Which personality traits influence exploratory testing?	Exploratory Testing	People having extrovert personality types are good exploratory testers.

[IMI10]	2010	Personal Characteristics	Students	What are the characteristics of high performing software testers in the industry?	high performing: effective in terms of the number of detected defects and by characteristics of testers seen important by managers and testers	experience, reflection, motivation and personal characteristics (thoroughness, conscientiousness, patience or persistency, accurateness, creativeness) were the top level themes; domain knowledge and specialized technical skills are more important than skills in test case desigh and test planning; the ability to reflect is an important characteristic of high performing domain expert tester

[OS10]	2010	MBTI	Students	What is an effective personality-type composition in software engineering teams?	Software quality was based on marks given by the evaluators. Teams that achieved 80 and above software quality marks were considered as effective teams	A balance of the personality types Sensing (S), Intuitive (N), Thinking (T) and Feeling (F) assisted teams in achieving higher software quality. In addition, Extroverted (E) members also had an impact on team performance.

cont.	because they achieved an excellent level in producing higher quality software.	A balance of the personality types Sensing (S), Intuitive (N), Thinking (T) and Feeling (F) assisted teams in achieving higher software quality. In addition, Extroverted (E) members also had an impact on team performance.

[SMGS10]	2010	Big Five	Students	Do differences in conscien-tiousness level affect the academic performance of students who pair programmed?	pair programming effectiveness was measured by academic performance and students' satisfaction	the academic performance of paired students was not significantly affectes by their level of conscientiousness; openness to experience showed the most prominent relationship with pair programming's effectiveness

[CdM+11]	2011	-	-	The aim of the literature review is to identify the methods used, topics addressed, personality tests applied, and the main findings produced in the research about software engineering.	-	Data extracted from 42 studies shows that pair programming and team building are the most recurring research topics and MBTI is the most used test.

[ACCA11]	2011	-	Practitioners / experts	Empirical assessment of different types of issue lead time prediction models using human factor measures collected from issue tracking systems	issue lead time (bug fixing time)	The number of stakeholders and average past issue lead time are important variables in constructing prediction models of issue lead time. Assignee experience, reporter experience, number of stakeholders, number of comments, average past lead time are human factors, that contribute significantly to issue lead time prediction.

[GFRV14]	2014	Big Five, among others	Students	How are the personal factors related to the female modeling performance and program-ming?	modeling examination task by two different SysML modeling tasks: anticipate a control problem described textually and graphically and to transfer it	German language, prior experience as well as several personal factors like self-concept, internality and social externality of success as well as conscientiousness and agreeableness are relevant factors influencing performance. Also frustration and the probability of

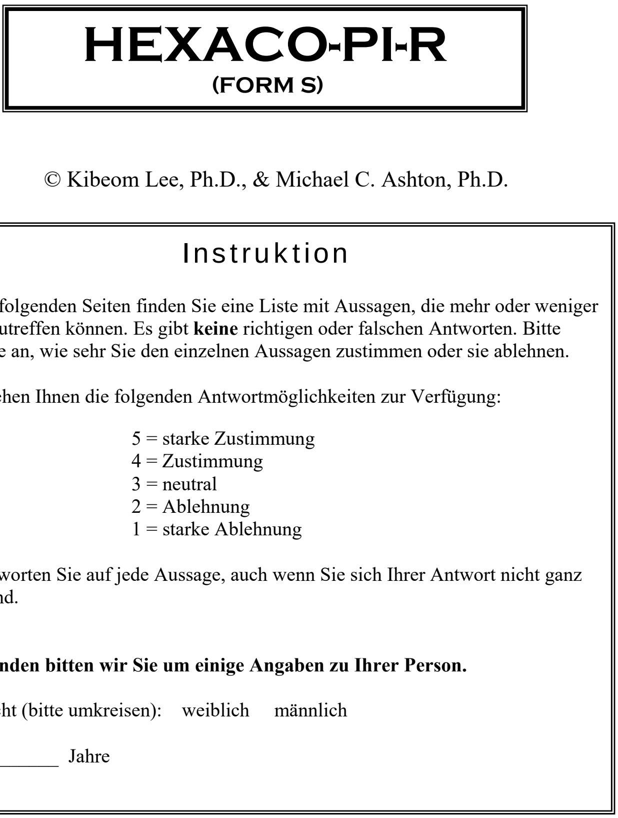
cont.	into a SysML representation; Programming performance was measured in C: the exercise contained the completion of gaps in a given code	success are associated with behavior modeling task performance. The investigation indicates that females have lower self-concept and ascribe their success less to their own abilities. Rather females showed higher social externality. This implies the

cont.	and the development of small code fragments; the overall performance in software engineering tasks was evaluated with the results of an exam	attribution of successful achievements to social environment rather to females' abilities. This may also cause the higher frustration in behavior modeling. In contrast, structure modeling is less affected by prior experience and personal factors. In

cont.	structure modeling predominantly cognitive abilities are necessary for females' performance. ... self-concept is negatively associated with performance. Females have lower self-beliefs but achieve good results in structure modeling. ... programming experience of females was only negatively associated with the probability of success. Therefore, especially females behavior modeling depends partly on personal factors.

Appendix B

Appendix B


HEXACO-PI-R
(FORM S)

© Kibeom Lee, Ph.D., & Michael C. Ashton, Ph.D.

Instruktion

Auf den folgenden Seiten finden Sie eine Liste mit Aussagen, die mehr oder weniger auf Sie zutreffen können. Es gibt **keine** richtigen oder falschen Antworten. Bitte geben Sie an, wie sehr Sie den einzelnen Aussagen zustimmen oder sie ablehnen.

Dafür stehen Ihnen die folgenden Antwortmöglichkeiten zur Verfügung:

5 = starke Zustimmung
4 = Zustimmung
3 = neutral
2 = Ablehnung
1 = starke Ablehnung

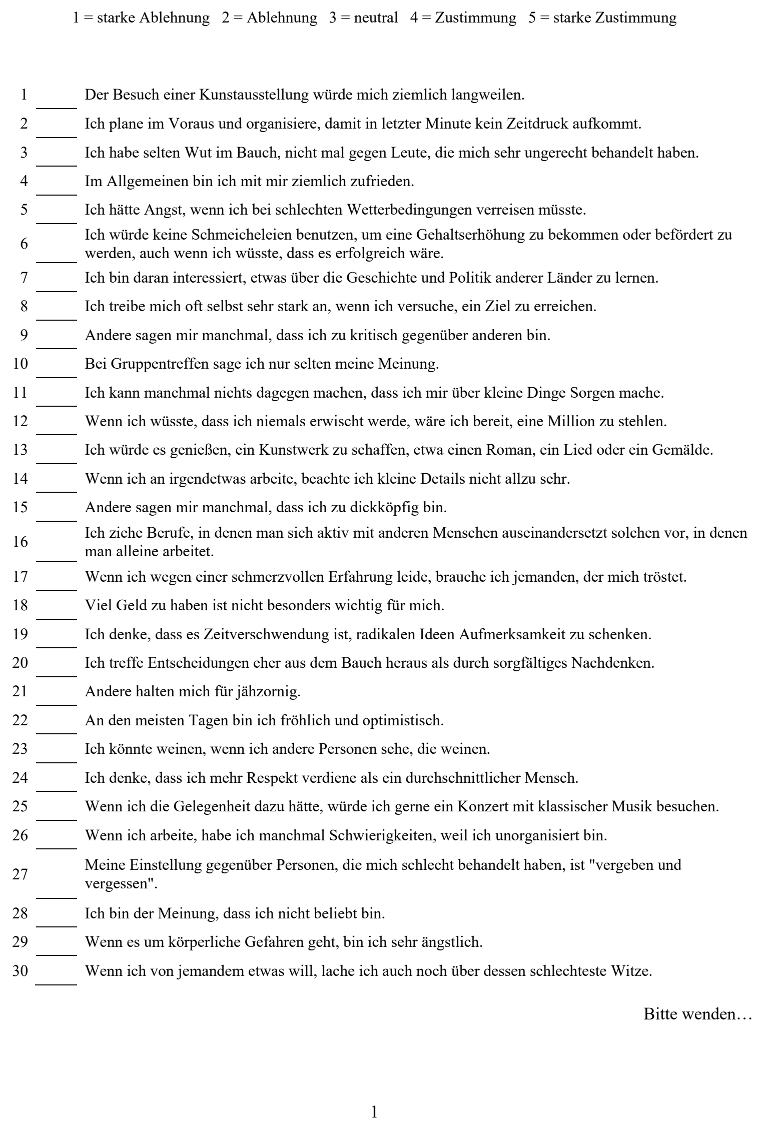
Bitte antworten Sie auf jede Aussage, auch wenn Sie sich Ihrer Antwort nicht ganz sicher sind.

Im folgenden bitten wir Sie um einige Angaben zu Ihrer Person.

Geschlecht (bitte umkreisen): weiblich männlich

Alter: _______ Jahre

Figure B.1.: German Version of the HEXACO-PI-R inventory with 60 questions, self-report version [LA09a].



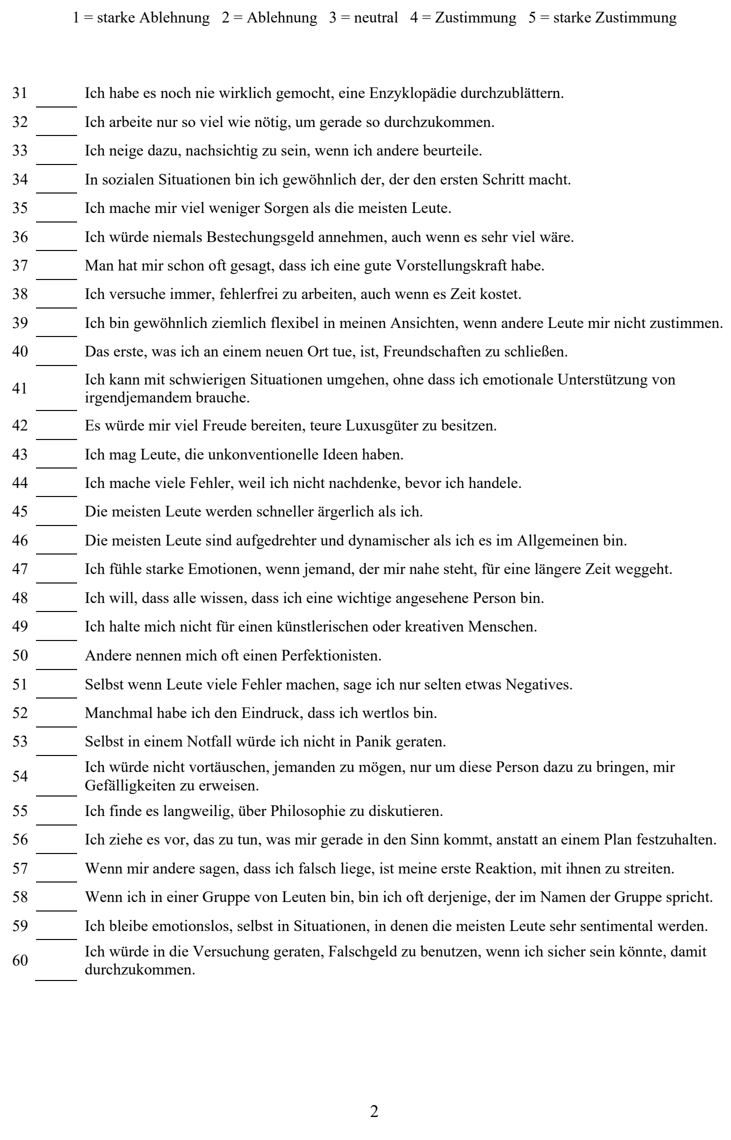
1 = starke Ablehnung   2 = Ablehnung   3 = neutral   4 = Zustimmung   5 = starke Zustimmung

1	_____	Der Besuch einer Kunstausstellung würde mich ziemlich langweilen.
2	_____	Ich plane im Voraus und organisiere, damit in letzter Minute kein Zeitdruck aufkommt.
3	_____	Ich habe selten Wut im Bauch, nicht mal gegen Leute, die mich sehr ungerecht behandelt haben.
4	_____	Im Allgemeinen bin ich mit mir ziemlich zufrieden.
5	_____	Ich hätte Angst, wenn ich bei schlechten Wetterbedingungen verreisen müsste.
6	_____	Ich würde keine Schmeicheleien benutzen, um eine Gehaltserhöhung zu bekommen oder befördert zu werden, auch wenn ich wüsste, dass es erfolgreich wäre.
7	_____	Ich bin daran interessiert, etwas über die Geschichte und Politik anderer Länder zu lernen.
8	_____	Ich treibe mich oft selbst sehr stark an, wenn ich versuche, ein Ziel zu erreichen.
9	_____	Andere sagen mir manchmal, dass ich zu kritisch gegenüber anderen bin.
10	_____	Bei Gruppentreffen sage ich nur selten meine Meinung.
11	_____	Ich kann manchmal nichts dagegen machen, dass ich mir über kleine Dinge Sorgen mache.
12	_____	Wenn ich wüsste, dass ich niemals erwischt werde, wäre ich bereit, eine Million zu stehlen.
13	_____	Ich würde es genießen, ein Kunstwerk zu schaffen, etwa einen Roman, ein Lied oder ein Gemälde.
14	_____	Wenn ich an irgendetwas arbeite, beachte ich kleine Details nicht allzu sehr.
15	_____	Andere sagen mir manchmal, dass ich zu dickköpfig bin.
16	_____	Ich ziehe Berufe, in denen man sich aktiv mit anderen Menschen auseinandersetzt solchen vor, in denen man alleine arbeitet.
17	_____	Wenn ich wegen einer schmerzvollen Erfahrung leide, brauche ich jemanden, der mich tröstet.
18	_____	Viel Geld zu haben ist nicht besonders wichtig für mich.
19	_____	Ich denke, dass es Zeitverschwendung ist, radikalen Ideen Aufmerksamkeit zu schenken.
20	_____	Ich treffe Entscheidungen eher aus dem Bauch heraus als durch sorgfältiges Nachdenken.
21	_____	Andere halten mich für jähzornig.
22	_____	An den meisten Tagen bin ich fröhlich und optimistisch.
23	_____	Ich könnte weinen, wenn ich andere Personen sehe, die weinen.
24	_____	Ich denke, dass ich mehr Respekt verdiene als ein durchschnittlicher Mensch.
25	_____	Wenn ich die Gelegenheit dazu hätte, würde ich gerne ein Konzert mit klassischer Musik besuchen.
26	_____	Wenn ich arbeite, habe ich manchmal Schwierigkeiten, weil ich unorganisiert bin.
27	_____	Meine Einstellung gegenüber Personen, die mich schlecht behandelt haben, ist "vergeben und vergessen".
28	_____	Ich bin der Meinung, dass ich nicht beliebt bin.
29	_____	Wenn es um körperliche Gefahren geht, bin ich sehr ängstlich.
30	_____	Wenn ich von jemandem etwas will, lache ich auch noch über dessen schlechteste Witze.

Bitte wenden…

1

German Version of the HEXACO-PI-R inventory with 60 questions, self-report version.


1 = starke Ablehnung   2 = Ablehnung   3 = neutral   4 = Zustimmung   5 = starke Zustimmung

31	_____	Ich habe es noch nie wirklich gemocht, eine Enzyklopädie durchzublättern.
32	_____	Ich arbeite nur so viel wie nötig, um gerade so durchzukommen.
33	_____	Ich neige dazu, nachsichtig zu sein, wenn ich andere beurteile.
34	_____	In sozialen Situationen bin ich gewöhnlich der, der den ersten Schritt macht.
35	_____	Ich mache mir viel weniger Sorgen als die meisten Leute.
36	_____	Ich würde niemals Bestechungsgeld annehmen, auch wenn es sehr viel wäre.
37	_____	Man hat mir schon oft gesagt, dass ich eine gute Vorstellungskraft habe.
38	_____	Ich versuche immer, fehlerfrei zu arbeiten, auch wenn es Zeit kostet.
39	_____	Ich bin gewöhnlich ziemlich flexibel in meinen Ansichten, wenn andere Leute mir nicht zustimmen.
40	_____	Das erste, was ich an einem neuen Ort tue, ist, Freundschaften zu schließen.
41	_____	Ich kann mit schwierigen Situationen umgehen, ohne dass ich emotionale Unterstützung von irgendjemandem brauche.
42	_____	Es würde mir viel Freude bereiten, teure Luxusgüter zu besitzen.
43	_____	Ich mag Leute, die unkonventionelle Ideen haben.
44	_____	Ich mache viele Fehler, weil ich nicht nachdenke, bevor ich handele.
45	_____	Die meisten Leute werden schneller ärgerlich als ich.
46	_____	Die meisten Leute sind aufgedrehter und dynamischer als ich es im Allgemeinen bin.
47	_____	Ich fühle starke Emotionen, wenn jemand, der mir nahe steht, für eine längere Zeit weggeht.
48	_____	Ich will, dass alle wissen, dass ich eine wichtige angesehene Person bin.
49	_____	Ich halte mich nicht für einen künstlerischen oder kreativen Menschen.
50	_____	Andere nennen mich oft einen Perfektionisten.
51	_____	Selbst wenn Leute viele Fehler machen, sage ich nur selten etwas Negatives.
52	_____	Manchmal habe ich den Eindruck, dass ich wertlos bin.
53	_____	Selbst in einem Notfall würde ich nicht in Panik geraten.
54	_____	Ich würde nicht vortäuschen, jemanden zu mögen, nur um diese Person dazu zu bringen, mir Gefälligkeiten zu erweisen.
55	_____	Ich finde es langweilig, über Philosophie zu diskutieren.
56	_____	Ich ziehe es vor, das zu tun, was mir gerade in den Sinn kommt, anstatt an einem Plan festzuhalten.
57	_____	Wenn mir andere sagen, dass ich falsch liege, ist meine erste Reaktion, mit ihnen zu streiten.
58	_____	Wenn ich in einer Gruppe von Leuten bin, bin ich oft derjenige, der im Namen der Gruppe spricht.
59	_____	Ich bleibe emotionslos, selbst in Situationen, in denen die meisten Leute sehr sentimental werden.
60	_____	Ich würde in die Versuchung geraten, Falschgeld zu benutzen, wenn ich sicher sein könnte, damit durchzukommen.

2

German Version of the HEXACO-PI-R inventory with 60 questions, self-report version.

Erklärung

Hiermit versichere ich, Erica Constanze Weilemann geb. Janke, dass ich die vorliegende Arbeit selbständig angefertigt habe und keine anderen als die angegebenen Quellen und Hilfsmittel und die Ratschläge von jeweils namentlich aufgeführten Personen benutzt sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe.

Erbach, 2020