UNIVERSITY OF GOTHENBURG

Effects of Personality and Expertise on Pair Programming

A comprehensive literature review on the effects of personality and expertise on pair programming, with the purpose to lay a foundation for how to configure pairs

Bachelor of Science Thesis in the Programme Software Engineering & Management

IOANNIS KELLARIS
PATRIK BÄCKSTRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, June 2015

Effects of Personality and Expertise on Pair Programming
A comprehensive literature review on the effects of personality and expertise on pair programming, with the purpose to lay a foundation for how to configure pairs

Ioannis Kellaris,
Patrik Bäckström,

Examiner: Michel R. V. Chaudron

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Effects of Personality and Expertise on Pair Programming

Ioannis Kellaris
Author
Software Engineering & Management, B.Sc
University of Gothenburg
jani.kellaris@gmail.com

Patrik Bäckström
Author
Software Engineering & Management, B.Sc
University of Gothenburg
mail@patrikbackstrom.com

Abstract

The purpose of this study is to suggest a method to support decision-makers in the configuration of optimal pairs, in order to increase pair programming effectiveness. To achieve this, findings of empirical studies regarding personality and expertise have been synthesised to create a mapping of the studied factors and their effects on pair programming. The mapping will be illustrated in a fictive setting, presenting how the synthesised findings can be applied in practice.

1 Introduction

Pair programming is a technique whereby code is produced by two software developers working on one machine interchangeably [1]. The benefits of pair programming are explored by many; in a meta-analysis of 18 studies on the effects of pairs versus individual developers, Hannay et al. report that pair programming has a positive effect on quality and duration, especially with regard to highly-complex tasks [5]. Hannay et al. also reported on the negative effect on effort, however Cockburn and Williams report that the increase in cost for the multiple benefits that pair programming introduces, including quality, duration, and learning amongst others is approximately 15% [3], far less than the expected 100% increase in cost.

The purpose of this study is to suggest a method for decision-makers to create optimal pairs, in order to maximise the potential of pair programming. To achieve this, we construct a "factor-effect" mapping based on the correlations of personality and expertise with pair programming, by conducting a comprehensive literature review on related empirical research. Furthermore, we illustrate the usage of this mapping in a fictive setting. The "factor-effect" mapping is intended to be the first step in proposing a method for configuring effective pairs.

While the benefits of pair programming have been extensively researched, we focus on the empirical studies that investigate factors that correlate with its effectiveness. We have identified that a multitude of empirical studies have been conducted to explore the effect of personality and expertise on pair programming, and that these effects can be significant. Sfetsos et al. report that design and code correctness are significantly correlated with pairs of diverse personalities and temperaments [F13, F15]. Arisholm et al. report that code correctness is increased for novice pairs compared to novice individuals [F1]. Thorbjorn and Hannay report that developers of similar Neuroticism [10] will verbalise more often [F16]; this act is considered crucial in successfully engaging novices in pair programming [14]. These factors and effects are an example of those that are explored in empirical research. However, due to differences in the approach and context of the existing empirical evidence, the problem is to synthesise the personality and expertise effects in a valuable approach.

The benefits that can be obtained by consulting the synthesised findings can have a practical effect on multiple aspects of pair programming. The value of our research is twofold. In *industry*, the aggregation of the factors and effects can result in an approach to improve the effectiveness of pair programming. In *academia*, we believe that a thoroughly reviewed background of the topic will aid the understanding of future researchers, and provide a reference point for further work.

Having identified personality and expertise as commonly researched areas in software development, we have formulated the following research questions to describe and classify the studied factors and effects (RQ1), and to illustrate an approach to apply the synthesised findings in practice (RQ2).

RQ1: Which factors are studied to explore the effects of personality and expertise on pair programming?

RQ2: How can the "factor-effect" mapping be applied in practice to define effective pairings for pair programming?

The rest of this report is structured as follows: Section 2 presents personality and expertise in psychology and software development, Section 3 presents our review methodology, Section 4 presents our results and discussion, Section 5 illustrates the application of the "factor-effect" mapping in a fictive setting, Section 6 presents the limitations of our study, and Section 7 presents our conclusions and future work.

2 Related work

This section introduces the personality models and their respective factors, which are addressed in the reviewed studies. Expertise is defined and described in the context of software development. This section also briefly discusses research on personality and software engineering teams.

2.1 Personality models

Cruz et al. conducted a systematic literature review, finding that the most common personality models that are utilised in software engineering are the Five-Factor Model (FFM), introduced by McCrae [10], the Myers-Briggs Type Indicator (MBTI) [11], and the Keirsey Temperament Sorter (KTS) [9]. These personality models are primarily used in the studies reviewed in this paper.

The FFM distils personality into 5 domains; Ogot and Okudan [12] concisely describe these domains: *Openness to experience* as the "number of interests one has and the extent to which they are pursued", *Neuroticism* as the "number and strength of stimuli that trigger negative emotions in a person", *Agreeableness* as the "number of sources from which one takes norms", *Extraversion* as the "number of relationships a person is comfortable with", and *Conscientiousness* as the "number of goals one is focused on". The NEO-PI questionnaire, based on FFM, is becoming popular among researchers, though MBTI largely dominates past studies regarding personality in software engineering [4].

The MBTI measures personality with a 94 item questionnaire, and is based on four bipolar discontinuous scales: *Introverting-Extraverting*, *Sensing-Intuiting*, *Thinking-Feeling*, and *Judging-Perceiving* [11]. A personality is classified as one of 16 personality types, based the largest score for each scale. KTS, measured by a 70 item questionnaire, classifies these 16 types into four temperament types [9]: *Artisan* (Sensing-Perceiving), *Guardian* (Sensing-Judging), *Idealist* (Intuiting-Feeling), *Rational* (Intuiting-Thinking).

2.2 Expertise in software engineering

Bryant [2] describes expertise as "a function of the four key elements of Practice, Strategy, Knowledge and Metacognition". Even though this concerns expertise in general, the same elements apply to pair programming [2]. Bryant's description of expertise resonates with the bases for identifying experts by Sim et al.: *years of experience*, *cognitive characteristics*, and *Software-Specific Proficiencies* (SSP).

In industry, *years of experience* is a common indicator, or delimiter, to ascertain the level of expertise. However, Sim et al. [15] found that SSP is the most prominent indicator of expertise. Sim et al. [15] present empirical evidence that individuals who are considered to be experts could be outperformed by a novice with *some SSP*, when introduced to areas unknown to the expert. Moreover, they state that an expert's "skill is often accompanied by a track record of accomplishments and consequently the achievements cannot be attributed to luck". This indicates that *years of experience* enables a software engineer to achieve a vast amount of knowledge over time, and therefore *years of experience* can be a determinant of *SSP*. Whereas, according to these findings, expertise should be determined in terms of the objectives and the nature of a given task, and not simply by *years of experience*, *SSP* can be complex to measure [15].

Furthermore, with respect to the cognitive characteristics of experts, Sim et al. [15] found that experts tend to have a systematic and orderly transition in the study of software components, compared to novices. The communication of experts was denoted by the quality of their explanations, that provided more details about the context of the problem, and that were less error-prone.

2.3 Personality and teams

Personality is also empirically studied in the context of software engineering teams; we present part of this body of research to denote the similarities and differences with studies on pair programming.

Pieterse and Kourrie [13] found that diversity in personality in teams is strongly correlated with team success. However, they also found that this correlation weakened over the course of a year, while the correlation of competence and team success strengthened. This interestingly signifies the notion that personality can be important in the initial stages of team formation, but that this importance diminishes over time.

Karn and Cowling [6] conducted ethnographic observations and analysed the effect of personality on student team performance. They found that teams with heterogeneity in personality, as well as ethnicity and religion worked well together. This resonates with the findings of reviewed studies synthesised in this research: diversity in personality has a positive effect on certain aspects of pair programming. However, it is not only diversity that is shown to keep a software engineering team cohesive and high-performing; Karn et al. [7] conclude that almost complete homogeneity results in high team cohesion. By extent, they also found that highly cohesive teams have a tendency to outperform teams with lower levels of cohesion, but also that high cohesion does not necessarily equal high performance.

3 Review methodology

In our research, we conduct a comprehensive literature review, with the goal to design a "factor-effect" mapping, to be consulted for the formation of effective pairs based on personality and expertise. Whereas the methodology of our study is in part based on Keele's guidelines for conducting an SLR [8], we do not consider our study to be a systematic literature review. In particular, the rigour of our search process, a prominent aspect of an SLR, was limited to two data sources, and we did not carry out a manual search process through all relevant journals [8]. Furthermore, limited quality criteria were considered in the data extraction process, focusing instead on the primary findings pertaining to our goal. However, we do identify that our study "can provide information about the effects of some phenomenon across a wide range of settings and empirical methods" [8].

3.1 Inclusion & Exclusion criteria

The inclusion criteria concerned empirical studies on personality or expertise effects on pair programming. More specifically, empirical studies that explored the effects of single personality factors, diverse personalities or expertise in the pair programming context were included. The eligibility of the studies was not restricted with respect to research design, nor did we impose any restrictions on whether studies were conducted at an academic or industrial setting. Furthermore, we included only those papers which were written in English.

Studies that were excluded did not concern pair programming in the context of personality or expertise. Studies that explored effects on teams were also excluded. Furthermore, we excluded those papers which solely presented anecdotal evidence.

3.2 Data source & search terms

The search terms were divided into two separate terms, each concerning one of our two areas of interest: personality and expertise. The total number of hits differed vastly between the two data sources (engineeringvillage.com and scopus.com) but yielded close to the same amount of relevant studies. Engineeringvillage has a narrow focus on engineering studies and therefore served our purpose well, regarding both personality and expertise in the pair programming context. Scopus' broader focus (Life-, Health-, Physical-, and Social Science, and Humanities) yielded significantly more hits, though however served as a complement when searching for studies regarding personality. Due to this broad focus, most of the studies were clearly aimed toward the aforementioned fields, not relating to personality or expertise in pair programming.

Table 1. Search terms & data sources

Search term	pair programming AND software AND personality	
Data source	engineeringvillage	scopus
Search date	3/17/2015	4/10/2015
Relevant studies	13	15
Total hits	62	276
Duplicates	12	
Selected studies	16	
Search term	pair programming AND software AND expert*	
Data source	engineeringvillage	scopus
Search date	3/17/2015	4/10/2015
Relevant studies	4	4
Total hits	44	897
Duplicates	4	
Selected studies	4	

3.3 Study selection

From our automatic search, 1279 papers were found. The procedure in evaluating the relevance of these papers initially involved examining the list of titles of the studies, excluding those which were evidently not related to our topic. The next step in this procedure was to read the abstract and conclusion of the remaining studies and assess their relevance by applying our inclusion and exclusion criteria. After this procedure, 20 papers were considered relevant.

Of these 20 papers upon closer inspection, 3 presented the same research and therefore duplicate information. For this reason, these studies were excluded from data extraction and synthesis.

In total, the findings of 17 empirical studies were extracted and synthesised, forming the "factor-effect" mapping (Section 4.1). The studies were given the identifiers F1-17.

3.4 Data extraction

The data extraction procedure initially involved developing the data extraction form, to systematically assess the papers under review. We based the structure of the data extraction form on Keele's guidelines [8]. We designed the form with the purpose to address RQ1: *Which factors are studied to explore the effects of personality and expertise on pair programming?* The data extraction form was pilot-tested by both researchers on a small number of studies, and was refined by excluding fields that were considered irrelevant or produced major inconsistencies in the end-result. The most important refinement was to establish a standard approach to mark the significance of the findings in the studies under review, since due to different research design and goals, 'significance' was not established by the same statistical tests. Therefore, instead of requiring information about the statistical analysis of the reviewed data, the form required marking the research findings with [ss] or [si], to denote 'statistically significant' or 'statistically insignificant'. The final form that was used for the extraction is presented in Table 2. The form was implemented in Google Sheets, due to the ease of access and remote collaboration possibilities that this platform provided.

Our data extraction procedure demanded one researcher to fulfil the role of data extractor (i.e. one who completed the data extraction), and the other researcher to function as the data checker (i.e. one who confirmed that the data extraction was carried out appropriately). The fulfilment of these roles by the researchers is suggested by Keele [8]. Our fulfilment of these roles differed, as the extractor initially performed a check on a subset of the papers, to verify that the extraction procedure was uniformly carried out by both researchers. However, at a later stage in our research, when synthesising the data, both researchers acted as data checkers, since certain information had to be approached in the specific context of the studies. In this respect, we identified that our data extraction form faced limitations, however, it still functioned as the primary source of information when conducting the data synthesis.

3.5 Synthesis

The process of synthesising the reviewed studies involved classifying and summarising the factors and effects of personality or expertise on pair programming. The application of the synthesised findings were illustrated in a fictive setting to answer RQ2: *How can the "factor-effect" mapping be applied in practice to define effective pairings for pair programming?*

To classify the findings, groups were constructed with the ultimate goal of our study in mind; depending on the desired outcome of a pair, the factors which affect the outcome can be considered and utilised independently. To emphasise on the expected outcome of the pair configurations, the synthesised data were ordered by the effect and prioritised by the degree of significance of the effect.

The quality of the effect was characterised by *increased, decreased,* or *no effect*. A concise definition of the effect was also included, as defined by the authors of the studies under review, to distinguish between the different meanings of similar or identical terminology that was used across research. The definitions of the effects that were not provided by the authors of the reviewed studies were respectively characterised by *Not defined by author(s)*.

The synthesised findings were tabulated into separate groups. To further clarify the factors, a concise narrative description also supplemented the "factor-effect" mapping. Figure 1 presents the steps taken to synthesis the findings.

Table 2. Data extraction form

ID	Item	Data
1	Data extractor	
2	Data checker	
3	Study title	
4	Study ID	
5	Number of institutions and/or companies addressed	
6	Number of participants	
7	Type of participants (professional/students)	
8	Personality model addressed in the study?	
9	Which factors are used in the research/study?	
10	What are the objectives of the study?	
11	Which research design is used?	
12	What are the effects on Pair Programming?[a]	
13	Authors' concerns about generalisability of results	
14	What validity threats are discussed?	

[a]Mark statistically significant findings with *[ss]*

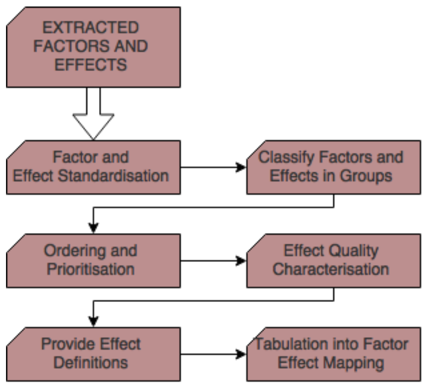
EXTRACTED FACTORS AND EFFECTS
Factor and Effect Standardisation
Classify Factors and Effects in Groups
Ordering and Prioritisation
Effect Quality Characterisation
Provide Effect Definitions
Tabulation into Factor Effect Mapping

Figure 1. Synthesis steps

4 Results and discussion

This section presents the synthesised findings from the reviewed studies, to address our research questions. We also present an overview of the empirical studies that present evidence of correlations. RQ1 is addressed by the "factor-effect" mapping. The findings presented in this section will furthermore be discussed and interpreted.

We identified that 17 empirical studies presented evidence (Table 3) for a correlation, or lack thereof, of personality or expertise with pair programming. Of the total studies, 11 were conducted at an academic setting, 5 were conducted at an industrial setting, and 1 study was conducted both in industry and academia. A small subset of these studies explored expertise in correlation with pair programming, of which 13 studies are on personality and 4 on expertise. This indicates that expertise in correlation to pair programming has not been extensively explored, strengthening the argument for the need for further work in this area, as is suggested in the reviewed literature on expertise.

The reviewed studies varied in objectives, with certain studies exploring the diversity in personality and expertise, or the effects of isolated personality traits, suggesting that these factors affect performance and other aspects of pair programming. These factors and effects are presented in Section 4.1. The reviewed studies are discussed in detail in Section 4.3.

4.1 Addressing RQ1

To address RQ1, *Which factors are studied to explore the effects of personality and expertise on pair programming?*, we synthesised the findings into the "factor-effect" mapping. The findings were grouped by effect to emphasise the usability of the "factor-effect" mapping. In essence, the mapping can be consulted by considering the desired effect, and determining which factors can lead to efficient pairs. The three groups that were constructed are: *Performance*, *Collaboration & Communication*, and *Learning*. The groups are respectively distinguished by whether the effects relate to the standard of the end product, the interactions and way-of-working of a pair, and the effects of personality on learning. Figure 2 shows the distribution of the reviewed studies per group.

Due to differences in the definitions of terminology in the studies, the effect definitions were included in "factor-effect" mapping to clarify ambiguities. Furthermore, the synthesised data was ordered by effect, and prioritised by the degree of significance of the effect. Therefore, if the desired outcome would be *Code Correctness*, all factors denoting that outcome are ordered by the effect's significance.

Some of the reviewed studies did not limit their objectives to solely exploring personality or expertise factors. We include these factors in the "factor-effect" mapping as well, since these are also studied in the respective studies' context, and the authors may conclude that these other factors can have a considerate effect on pair programming. We also consider that this adds value to the mapping, and may provide the grounds for future research not limited to personality or expertise.

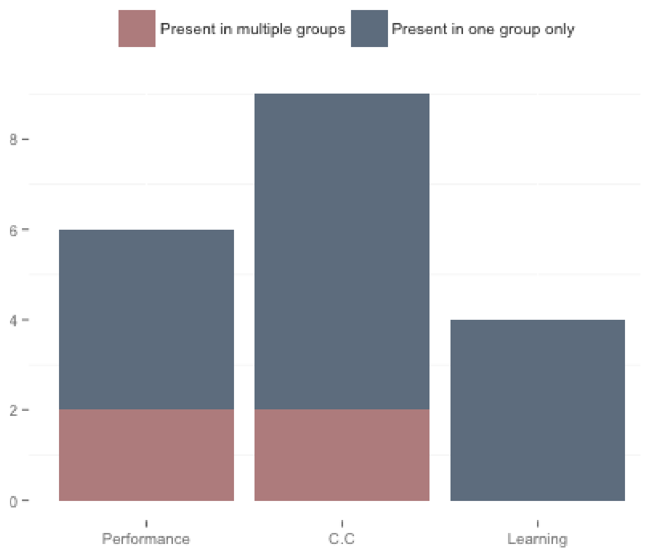

Figure 2. Distribution of the reviewed studies per group

Figure 3 depicts the synthesised factors, effects, and their number of occurrences. The factors and effects of the learning group are not depicted in this figure, and are presented in section 4.1.3. This chart makes it evident that most synthesised findings relate to diversity, either in general on in particular personality traits.

Figure 4 presents all factors in the performance and C&C groups, categorised based on theme (personality, expertise, other). Furthermore, this figure presents the factors that have an increased effect on the left, and those that have a decreased effect or no effect on the right. The corresponding effects can be found in Table 4 and Table 5.

4.1.1 Performance group

The *Performance* group includes those effects in the reviewed studies that relate to the standard of the end-product, primarily: code correctness, code productivity, code quality, design correctness, and velocity.

Diverse personalities are always found to be significantly correlated with increased performance of a pair, as evidenced by F4, F13, F15. Studies F13 and F15 take into

Table 3. Reviewed studies

ID	Title	Model	Setting	Participants
[F1]	Evaluating Pair Programming With Respect to System Complexity and Programmer Expertise	NA	Industrial	295
[F2]	Double Trouble: Mixing Qualitative and Quantitative Methods in the Study of Extreme Programmers	NA	Industrial	14 pairs
[F3]	Critical Personality Traits in Successful Pair Programming	UD[a]	Both	118
[F4]	Exploring the Underlying Aspects of Pair Programming: The Impact of Personality	MBTI	Academic	128
[F5]	Pair Dynamics in Team Collaboration	MBTI	Academic	128
[F6]	The Social Dynamics of Pair Programming	NA	Industrial	19
[F7]	Effects of Personality on Pair Programming	FFM	Industrial	196
[F8]	Pair Programming Productivity: novice–novice vs expert–expert	NA	Academic	40
[F9]	The Effects of Openness to Experience on Pair Programming in a Higher Education Context	FFM	Academic	137
[F10]	An Empirical Study of the Effects of Personality in Pair Programming Using the Five-Factor Model	FFM	Academic	49
[F12]	The Effects of Neuroticism on Pair Programming: An Empirical Study in the Higher Education Context	FFM	Academic	118
[F11]	An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-Factor Personality Model	FFM	Academic	218
[F13]	Investigating the Impact of Personality and Temperament Traits on Pair Programming: A Controlled Experiment Replication	MBTI/KTS	Academic	160
[F14]	Investigating the Impact of Personality Types on Communication and Collaboration-viability in Pair Programming: An Empirical Study	MBTI/KTS	Academic	84
[F15]	An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming	MBTI/KTS	Academic	70
[F16]	Personality and the Nature of Collaboration in Pair Programming	FFM	Industrial	44
[F17]	Examining the Compatibility of Student Pair Programmers	MBTI	Academic	1350

[a]University of Denver Career Center: http://www.du.edu/career/media/documents/pdfs/personality.pdf, as of May 2015

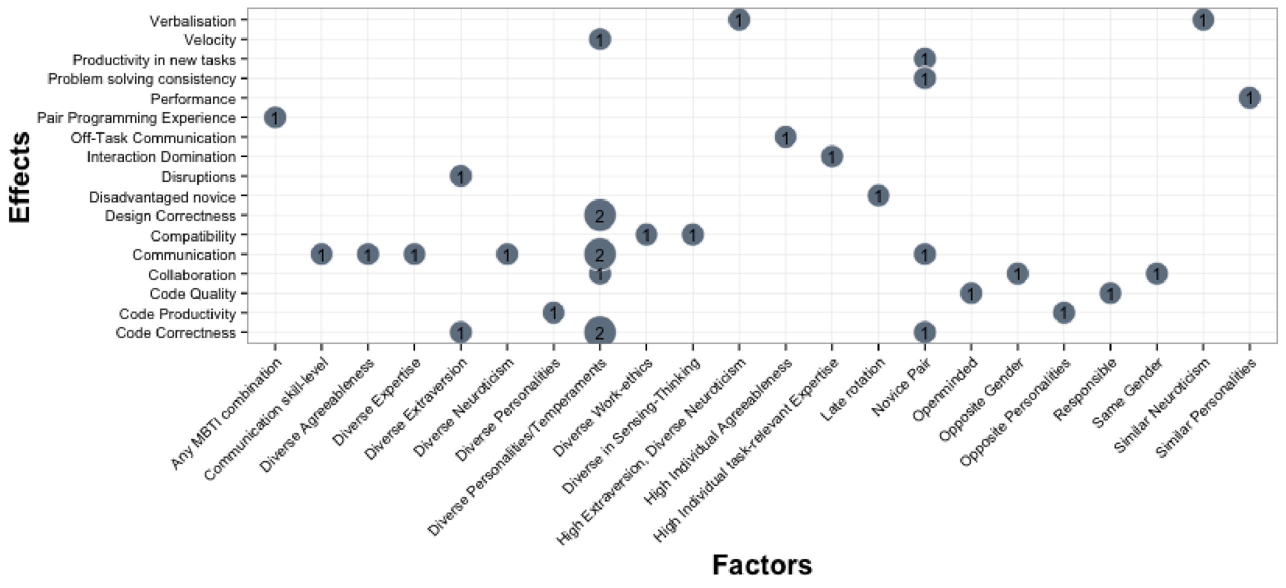

Figure 3. Factors and effects found in the reviewed literature

Table 4. Performance group

ID	Factor	Level	Effect	Effect definition	SS[a]
[F13]	Diverse Personalities/Temperaments	Increased	Code Correctness	Not defined by author(s)	Y
[F15]	Diverse Personalities/Temperaments	Increased	Code Correctness	——"——	Y
[F1]	Novice Pair	Increased[b]	Code Correctness	Intended functionality is properly implemented	N
[F4]	Diverse Personalities	Increased[c]	Code Productivity	Executable code and coding effort	Y
[F4]	Opposite Personalities	Increased[c]	Code Productivity	——"——	N
[F3]	Openminded[d]	Increased	Code Quality	Output correctness, required documentation, good programming style, correct use of objects, interface design	N
[F3]	Responsible[e]	Increased	Code Quality	——"——	N
[F13]	Diverse Personalities/Temperaments	Increased	Design Correctness	Not defined by author(s)	Y
[F15]	Diverse Personalities/Temperaments	Increased	Design Correctness	——"——	Y
[F7]	Diverse Extraversion	Increased	Performance	Code Correctness, Duration, Methodology, Extensibility, Cost Effectiveness, Redesign, Regression Grade	Y
[F4]	Similar Personalities	Decreased[c]	Performance	Executable code and coding effort	N
[F8]	Novice Pair	Increased[b]	Productivity in new tasks	Velocity and quality	NA
[F15]	Diverse Personalities/Temperaments	Increased	Velocity	Less time to complete each task	Y

[a]Statistically significant correlation
[b]In comparison with a novice individual
[c]**Level** of *Diverse* and *Opposite* in comparison to *Similar*
[d]Defined as *impartial, receptive to new ideas*, and *free from prejudice*
[e]Defined as *dependable, trustworthy, reliable*, and *loyal*

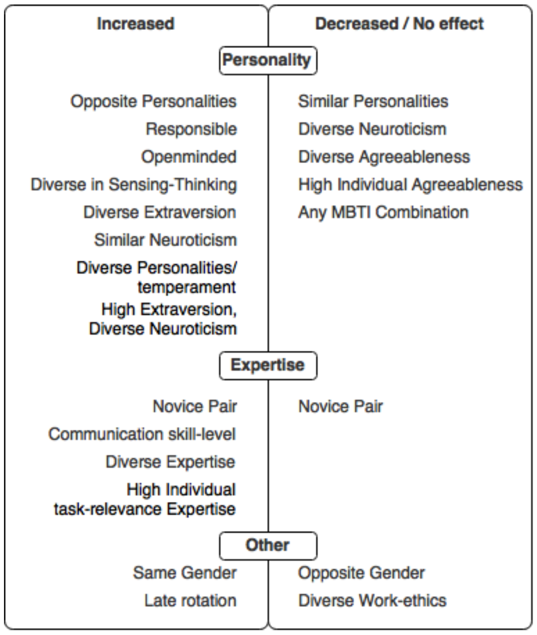

Figure 4. Factor grouping

account temperament, along with personality. Study F7 presents a finding with respect to diversity in a particular trait, as opposed to diversity in general; specifically, *Diverse Extraversion* has a significant correlation with performance.

Similar personalities, addressed by F4, are shown to correlate to decreased performance. *Opposite personalities* have a correlation to increased performance compared to *Similar personalities*, however, with no statistical significance between *Opposite* and *Similar*. There is, however, a significant difference between the performance of *Diverse personalities* opposed to *Opposite* and *Similar*, as addressed by F4.

Study F3 shows that a correlation of *Openminded* and *Responsible* with code quality exists, however, with no statistical significance. In this study, the findings supported that at least one developer with high *Openminded* or *Responsible* would result in increased code quality. It is noteworthy that this study did not measure personality with the models utilised in other studies, and therefore these factors are unique in the “factor-effect” mapping.

With respect to expertise, F8 correlates *Novice Pairs* with increased productivity in new tasks, in terms of velocity and quality, compared to novice individuals. Study F1 shows that a *Novice pair* correlates with increased performance, compared to novice individuals, and shows to have similar performance with an expert individual. However, this finding has no statistical significance.

Table 5. Collaboration & Communication group

ID	Factor	Level	Effect	Effect definition	SS[a]
[F14]	Diverse Personalities/Temperament	Increased	Collaboration	Developers' satisfaction, knowledge acquisition, participation	Y
[F5]	Same Gender	Increased	Collaboration	Higher level of communication, satisfaction, compatibility, confidence	Y
[F5]	Opposite Gender	Decreased	Collaboration	Lower level of communication, satisfaction, compatibility, confidence	Y
[F13]	Diverse Personalities/Temperament	Increased	Communication	Number of communication transactions made during the pair programming session	Y
[F5]	Communication skill-level	Increased	Communication	Not defined by author(s)	Y
[F15]	Diverse Personalities/Temperament	Increased	Communication	Number of communication transactions made during the pair programming session	Y
[F16]	Diverse Neuroticism	No effect	Communication[b]	Both developers contribute new information to a task	N
[F16]	Diverse Agreeableness	Decreased	Communication	——"——	N
[F2]	Novice Pair	Increased	Communication	Not defined by author(s)	NA
[F2]	Diverse Expertise	Increased	Communication	More rest and reminders of errors needing solutions	NA
[F17]	Diverse in Sensing-Thinking	Increased	Compatibility	Not defined by author(s)	Y
[F17]	Diverse Work-ethics	Decreased	Compatibility	——"——	Y
[F6]	Late rotation	Increased	Disadvantaged novice	Inhibits novice programmers to contribute effectively	NA
[F16]	Diverse Extraversion	Increased	Disruptions[c]	If a developer changes the topic, however, it is not defined whether the new topic relates to the task	Y
[F6]	High Individual task-relevant Expertise	Increased	Interaction Domination	Determining how and what to implement	NA
[F16]	High Individual Agreeableness	Decreased	Off-Task Communication	Small-talk, irrelevant to solving a task	Y
[F5]	Any MBTI combination	No effect	Pair Programming Experience	In the levels of satisfaction, compatibility, communication, and confidence	N
[F2]	Novice Pair	Decreased	Problem solving consistency	Inconsistent approaches to solving problems	NA
[F16]	High Extraversion, Diverse Neuroticism	Increased	Verbalisation	Programming aloud	NA
[F16]	Similar Neuroticism	Increased	Verbalisation	——"——	NA

[a]Statistically significant correlation
[b]Neuroticism does not increase the amount of communication-intensive collaboration
[c]Increased amount of disruptions compared with Introverts. Disruptions can be productive or counter-productive, on-topic or off-topic

4.1.2 Collaboration & Communication group

The *Collaboration & Communication* group includes those effects in the reviewed studies that relate to the interactions and way-of-working of a pair. This group primarily includes effects on: collaboration, communication, compatibility, and verbalisation.

Similar to the *Performance* group, *Diverse Personalities and Temperaments* have a significant correlation with collaboration and communication as evidenced by F13, F14, and F15. Study F17 finds that a pair of developers with *Sensing-Thinking* traits have increased compatibility.

Diversity in certain personality traits is, however, shown to correlate with decreased communication; F16 demonstrates this about *Diverse Agreeableness*, though with no statistical significance. Study F16 also presents evidence that one individual of *High Agreeableness* will introduce more off-task communication to the pair, and that a pair with *Diverse Extraversion* will experience more disruptions, however, the study does not distinguish whether or not these disruptions are task-related. *High Extraversion* and *Diverse Neuroticism* correlates with an increase in verbalisations. Diversity in *Neuroticism* has otherwise no effect on Communication. Conversely, *Similar Neuroticism* correlates with an increase in verbalisation.

Study F5 contradicts other findings, concluding that personality, and specifically *Any MBTI combination*, has no effect on the pair programming experience (including compatibility and communication). This study also investigates the impact of *Communication skill-level* on pair programming, finding that its only effect is on the aspect of communication. Furthermore, it investigates *Gender* as a factor, which correlates with collaboration; *Same Gender* pairs experience increased collaboration, whereas for *Opposite Gender* pairs collaboration is decreased. Other studies that explore factors beyond personality and expertise are F6 and F17. The findings, respectively, are that a pair with *Diverse Work-ethics* has decreased compatibility, and that *Late ro-*

tation (i.e. switching partners late in a project) can disadvantage a novice.

Regarding expertise, F2 finds that *Novice Pairs* have decreased problem solving consistency, and increased communication. Pairs of *Diverse Expertise* are also correlated with increased communication. Developers with *High Individual task-relevant Expertise* are evidenced to dominate interaction, according to F6.

4.1.3 Learning group

The *Learning* group concerns the effects of personality on learning. The findings in this group are represented only textually, due to the particular objectives and design of the studies. In particular, the objective of F9, F10, F11, and F12 is to assess the effectiveness of pair programming as a pedagogical tool in academia. These studies share a common experiment design, whereby students are paired based on certain personality traits, and are then assessed *individually* to determine to which extent pair programming has affected the learning outcomes.

The consensus of F9, F10, and F12 is that *Openness to experience* significantly correlates with students' academic performance. *Conscientiousness* is not found to correlate with academic performance in F11 and F12, but a non-significant correlation is found in F10. Finally, *Neuroticism* does not have an impact on academic performance, according to F11.

4.2 Addressing RQ2

RQ2 is addressed in Section 5, by illustrating how the "factor-effect" mapping may be applied to define effective pairs in a fictive setting. We address RQ2 after discussing the "factor-effect" mapping to provide the reader with a better understanding of the implications of the findings.

4.3 Discussion of the "factor-effect" mapping

Overall, the strongest predictor of increased performance, collaboration and communication is diversity in personality. Particularly, all findings in the *Performance* group that concern diversity in personality are statistically significant. The immediate effects are on code correctness, design correctness, productivity, and velocity, which are aspects that any organisation can benefit from in their software development process. In the *C&C* group, 5 out of 7 findings that concern diversity are also statistically significant. This indicates that an increase in the aspect of collaboration and communication may positively affect developers' satisfaction, knowledge acquisition and participation, as well as increase their communication transactions. Diverse personalities may also be more prone to disruptions, however, it is possible that these disruptions are part of a task-relevant discussion that benefits their productivity. Therefore, a decision maker, in the process of configuring effective pairs, should prioritise pairing those developers that present some diversity in their personality, as opposed to pairing similar personalities.

It appears that diverse personality pairs can increase the effectiveness of pair programming, however, diversity in *Agreeableness* and *Neuroticism* are not found to have a significant correlation. *Similar Neuroticism* in a pair, as well as the combination of *High Extraversion* and *Diverse Neuroticism* are shown to correlate with an increase in verbalisation which can positively engage a novice [14]. This indicates that, even though *Neuroticism* itself does not correlate to increased effectiveness, it can still be considered along with other factors.

One study presents evidence for no correlation of personality with the levels of satisfaction and communication (*Any MBTI combination* in Table 5) contradicting 4 studies that report a significant correlation of personality with these effects. It is interesting, however, that this study explores the effects of gender, and finds that *Same Gender* pairs have significantly increased collaboration in terms of communication, satisfaction, compatibility, and confidence, compared to *Opposite Gender* pairs. One other factor that can significantly affect compatibility is work-ethics; in particular, one study that explores its effects reports that *Diverse work-ethics* leads to decreased pair compatibility. Contradictorily to diversity in personality, diversity in work-ethics and mixed genders correlate with a decrease in certain aspects of pair collaboration.

With regard to learning, the studies that assess pair programming as a pedagogical tool find that *Openness to experience* is the most significant predictor for student performance. In particular, this finding is statistically significant in 3 out of 3 studies that explore the effect of this trait. A significant correlation is also found for *Conscientiousness*, but in only 1 out of 3 studies. Neuroticism is not found to be a good predictor for student performance either. The focus of these studies, however, is not how these traits affect pair programming, but instead how these traits correlate to learning outcomes of individuals.

The effects of expertise on performance are explored by two reviewed studies, which report that a *Novice pair* correlates with increased code correctness, quality, and velocity compared to a novice individual. A novice pair is also found to have increased communication, and decreased problem solving consistency. In pairs of *Diverse Expertise*, communication is also increased. This clearly indicates that a novice benefits from increased communication transactions, due to lack of knowledge and need for guidance in pair programming. Furthermore, a developer's *High task-relevant Expertise* is found to correlate with increased interaction domination, a finding that illustrates that the disengagement of a developer could occur despite their level of expertise in general; this may suggest that for tasks that demand highly relevant expertise, it would be preferable that two developers with similar knowledge are paired.

5 Illustration of the application of the "factor-effect" mapping in a fictive setting

The purpose of the illustration is to address RQ2: *How can the "factor-effect" mapping be applied in practice to define effective pairings for pair programming?* We present the application of the "factor-effect" mapping in a fictive industrial setting, in order to illustrate how the mapping can be used in practice. The basis of this illustration can be used in constructing an empirical study in future, as it demonstrates the data collection procedure, utilisation of the "factor-effect" mapping, and the evaluation of its effectiveness.

For this illustration, we collected personality profiles from students at the Software Engineering & Management programme at the University of Gothenburg. The questionnaire that was used is the short form of the IPIP-NEO[a], a 120 item questionnaire based on the Five-Factor Model. Student profiles were collected for reasons of convenience, and we received 6 responses (Table 6) over a six week period. Since the expertise of students was not considered relevant, the profiles were supplemented by devising their expertise level in order to highlight certain aspects of the "factor-effect" mapping.

5.1 Fictive industrial setting

The industrial setting is a telecommunications company, specialising in security critical software development, employing 20 developers. They have, in recent months, introduced pair programming to increase the end-product quality by reducing post-release bugs, and to improve developers' productivity. They have been pilot-testing pair programming with certain developers, however, the company has not yet evaluated pair programming to be a successful change. They have established metrics for measuring code and design correctness, and velocity, as well as developer satisfaction with relevant questionnaires. They are therefore at the stage of re-approaching the formation of the pilot-testing group, and are interested in applying the "factor-effect" mapping, as they see it as a promising tool for the formation of effective pairs.

5.2 Participants

For the renewed initiative, the 20 developers took the IPIP-NEO[a] personality test, and their expertise has been determined by their software specific proficiencies that are relevant to the upcoming tasks, as well as the years of programming experience. The knowledge of the developers about the upcoming task is gauged by a short programming test on recurring problems in the particular domain[b]. The decision-makers at the company are determined to increase the number of specialists in security critical software development, and are therefore also interested in the knowledge acquisition of novice developers. To this end, it was decided that 3 novices and 3 experts would be selected as the pilot team to assess the extent to which the "factor-effect" mapping can increase performance, collaboration, and learning.

The personality profiles of the participants are presented in Table 6, and their devised level of expertise is denoted by "E" (Expert) and "N" (Novice).

5.3 Pair configuration

Table 6. Developer profiles

ID	E[a]	A[a]	C[a]	N[a]	O[a]
N1	L	H	H	L	A
N2	L	H	A	A	A
N3	A	L	A	A	A
E1	A	H	H	L	H
E2	A	L	H	L	L
E3	A	A	L	A	A

[a]**E:** Extraversion, **A:** Agreeableness, **C:** Conscientiousness, **N:** Neuroticism, **O:** Openness to experience
[b]**L:** Low, **A:** Average, **H:** High

Developers N2 and E3 were paired so that N2 can specialise in security critical tasks. By consulting the "factor-effect" mapping, the decision-makers decided to pair N2 and E3 based on the following factors: *Similar Neuroticism* and *Diverse Expertise* (Table 5). The desired effect is to increase verbalisation, and to maintain the quality of the outcome, through reminders of errors on behalf of the expert. This particular novice, due to their low extraversion, can particularly benefit from acting as the driver and verbalising throughout the task, both crucial aspects for the engagement of a novice [14].

Developers E1 and E2 were paired to undertake an important security critical task that requires *High Individual task-relevant Expertise* (Table 5). Due to the nature of the task, the decision-makers consulted the "factor-effect" mapping and decided that due to E1's *High Individual Agree-*

[a]http://www.personal.psu.edu/faculty/j/5/j5j/IPIP/ipipneo120.htm, as of May 2015
[b]Due to the challenging nature of determining expertise, a specific test cannot be referenced. Gauging for expertise is highly dependent on the development context and expected outcome, and is challenging to standardise [15]

ableness (Table 5), and their diversity with E2's low individual agreeableness, that a decrease in off-task communication would be achieved. This suggests that the developers will have fewer, but task-related discussions. Their consistency in problem solving, as experts, should yield a high quality result, despite limited diversity in personality.

Developers N1 and N3 were paired for non-security critical tasks, as they have low software specific proficiencies in this field. However, their task demands a high-quality result, and were therefore paired based on their *Diverse Personalities* (Table 4) in order to increase performance. Their average score on the trait *Openness to experience* indicates that they are good candidates for learning (Section 4.1.3). Their expected decreased problem solving consistency may be moderated by their *Diverse Extraversion* (Table 5), which can trigger the expression of ideas (in the form of positive disruptions) that can help them to explore alternative solutions.

5.4 Industrial evaluation of the "factor-effect" mapping

After a month, the resulting work of the developers is measured in terms of performance and collaboration with the company's established metrics. The learning outcomes of the novice-novice group is determined by their newly acquired software specific proficiencies[b]. By evaluating the results, the decision-makers can conclude whether the results are promising or not, and expand the initiative to include all developers.

6 Limitations

One important limitation for our review methodology is that the search strategy included two sources, and no manual search for studies in related journals. Considering the amount of duplicate results we found in the second source, it is likely that further sources would not yield unique results. Furthermore, the studies concerning expertise are 4, a small subset of the reviewed studies, therefore our primary findings mostly concern personality. The limitation in this respect is that the "factor-effect" mapping does not include as many entries on expertise, but this should not be viewed as a limitation to the importance of expertise in pair programming. It rather suggests that expertise should be further explored in future research.

Regarding our data extraction, limited quality criteria were applied, so we did not account for the potential impact of the findings attributed to the quality of the reviewed empirical research. Instead, we focused on that information that was relevant to address our research questions. Another limitation of our data extraction procedure is that we did not extract statistical significance values for the findings. This was due to the complexity and differences between the statistical analyses that were used in the reviewed studies, but we denoted those findings that had significant correlations with a binary value (Y/N). In the synthesis, a possible limitation can be attributed to misinterpretations due to lack of definitions in reviewed studies. However, to address this concern, definitions of the effects were included in the "factor-effect" mapping for the studies that supplied them.

The personality profiles that were utilised in the illustration of the "factor-effect" mapping were students'. It is not within the scope of this study to assess whether personality in industry differs from that of students, however, we do not consider this an important limitation, as the purpose of this illustration was to suggest how the "factor-effect" mapping can be applied in practice. Finally, in the illustration, we address diversity as differences in the Five-Factor Model traits, without determining how much of a difference is considered diverse enough.

7 Conclusions and future work

This comprehensive literature review analysed 17 empirical studies on the effects of personality and expertise on pair programming, with the objective to suggest a method for decision-makers to create optimal pairs. The factors and effects were extracted from the studies and synthesised into the "factor-effect" mapping. Furthermore, we illustrated the usage of the mapping in a fictive setting, describing one possible approach on how to apply it in industry.

The reviewed studies explored multiple factors, primarily diversity in personality, and their effects on pair programming performance, developer interaction, and learning. The most prominent factor, diversity in personality in general or in particular personality traits, is found to be significantly correlated with pair programming performance, collaboration and communication in 12 out of 15 findings. Our conclusion is that diversity in personality strongly correlates with performance, collaboration, and communication facets. The studies that explored the effectiveness of pair programming as a pedagogical tool reported that *Openness to experience* is the most significant predictor of academic performance. With regard to expertise, the most common findings relate to developers' communication and collaboration.

For the advancement of our research, we propose that the "factor-effect" mapping is evaluated in practice at an industrial environment, where expertise is of relevance, as opposed to academia. An experiment with students is considered valuable as well, however in this case, only the ef-

[b]Due to the challenging nature of determining expertise, a specific test cannot be referenced. Gauging for expertise is highly dependent on the development context and expected outcome, and is challenging to standardise [15]

fects of personality could be explored in a meaningful way. For the measurement of personality, we encourage future researchers to consider whether the MBTI or the FFM, or possibly other personality models, are most accurate in determining diversity in personality. Furthermore, as part of the evaluation, an appropriate test for expertise should be constructed; our suggestion is that software-specific proficiencies are taken into account, based on Sim et al. [15].

Finally, this research should be replicated and extended, to include relevant factors and effects from future empirical research.

Acknowledgements

We thank our supervisors, Per Lenberg, for his valuable feedback, and Jan-Philipp Steghöfer, for insightful advice and inspiration.

References

[1] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

[2] S. Bryant. Rating Expertise in Collaborative Software Development. In *Proc. PPIG*, pages 19–29, 2005.

[3] A. Cockburn and L. Williams. The Costs and Benefits of Pair Programming. *Extreme programming examined*, pages 223–247, 2000.

[4] S. Cruz, F. Q. da Silva, C. V. Monteiro, C. Santos, and M. Dos Santos. Personality in Software Engineering: Preliminary Findings From a Systematic Literature Review. In *Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, pages 1–10. IET, 2011.

[5] J. E. Hannay, T. Dybå, E. Arisholm, and D. I. Sjøberg. The Effectiveness of Pair Programming: A Meta-analysis. *Information and Software Technology*, 51(7):1110–1122, 2009.

[6] J. Karn and T. Cowling. A follow up study of the effect of personality on the performance of software engineering teams. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 232–241. ACM, 2006.

[7] J. Karn, S. Syed-Abdullah, A. J. Cowling, and M. Holcombe. A study into the effects of personality type and methodology on cohesion in software engineering teams. *Behaviour & Information Technology*, 26(2):99–111, 2007.

[8] S. Keele. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical report, Technical report, EBSE Technical Report EBSE-2007-01, 2007.

[9] D. Keirsey. The Keirsey Temperament Sorter. *Available online at http://keirsey. com*, 1984.

[10] R. R. McCrae and O. P. John. An Introduction to the Five-Factor Model and its Applications. *Journal of personality*, 60(2):175–215, 1992.

[11] I. B. Myers. The Myers-Briggs Type Indicator: Manual (1962). 1962.

[12] M. Ogot and G. E. Okudan. The Five-Factor Model Personality Assessment for Improved Student Design Team Performance. *European Journal of Engineering Education*, 31(5):517–529, 2006.

[13] V. Pieterse, D. G. Kourie, and 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*, pages 180–186. South African Institute for Computer Scientists and Information Technologists, 2006.

[14] L. Plonka, H. Sharp, and J. van der Linden. Disengagement in Pair Programming: Does it Matter? In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 496–506. IEEE, 2012.

[15] S. E. Sim, S. Ratanotayanon, O. Aiyelokun, and E. Morris. An Initial Study to Develop an Empirical Test for Software Engineering Expertise. *Institute for Software Research, University of California, Irvine, Irvine, CA, USA, Technical Report# UCI-ISR-06-6*, 2006.

Studies selected for review

[F1] E. Arisholm, H. Gallis, T. Dyba, and D. I. Sjoberg. Evaluating Pair Programming With Respect to System Complexity and Programmer Expertise. *Software Engineering, IEEE Transactions on*, 33(2):65–86, 2007.

[F2] S. Bryant. Double Trouble: Mixing Qualitative and Quantitative Methods in the Study of Extreme Programmers. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pages 55–61. IEEE, 2004.

[F3] J. Chao and G. Atli. Critical Personality Traits in Successful Pair Programming. In *Agile Conference, 2006*, pages 5–pp. IEEE, 2006.

[F4] K. S. Choi, F. P. Deek, and I. Im. Exploring the Underlying Aspects of Pair Programming: The Impact of Personality. *Information and Software Technology*, 50(11):1114–1126, 2008.

[F5] K. S. Choi, F. P. Deek, and I. Im. Pair Dynamics in Team Collaboration. *Computers in Human Behavior*, 25(4):844–852, 2009.

[F6] J. Chong and T. Hurlbutt. The Social Dynamics of Pair Programming. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 354–363. IEEE, 2007.

[F7] J. E. Hannay, E. Arisholm, H. Engvik, and D. I. Sjoberg. Effects of Personality on Pair Programming. *Software Engineering, IEEE Transactions on*, 36(1):61–80, 2010.

[F8] K. M. Lui and K. C. Chan. Pair Programming Productivity: Novice–Novice vs. Expert–Expert. *International Journal of Human-computer studies*, 64(9):915–925, 2006.

[F9] N. Salleh, E. Mendes, and J. Grundy. The Effects of Openness to Experience on Pair Programming in a Higher Education Context. In *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, pages 149–158. IEEE, 2011.

[F10] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch. An Empirical Study of the Effects of Personality in Pair Programming Using the Five-Factor Model. In *Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement*, pages 214–225. IEEE Computer Society, 2009.

[F11] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch. An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-Factor Personality Model. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 577–586. ACM, 2010.

[F12] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch. The Effects of Neuroticism on Pair Programming: An Empirical Study in the Higher Education Context. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement*, page 22. ACM, 2010.

[F13] P. Sfetsos, P. Adamidis, L. Angelis, I. Stamelos, and I. Deligiannis. Investigating the Impact of Personality and Temperament Traits on Pair Programming: A Controlled Experiment Replication. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pages 57–65. IEEE, 2012.

[F14] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis. Investigating the Impact of Personality Types on Communication and Collaboration-viability in Pair Programming: An Empirical Study. In *Extreme Programming and Agile Processes in Software Engineering*, pages 43–52. Springer, 2006.

[F15] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis. An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming. *Empirical Software Engineering*, 14(2):187–226, 2009.

[F16] T. Walle and J. E. Hannay. Personality and the Nature of Collaboration in Pair Programming. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 203–213. IEEE, 2009.

[F17] L. Williams, L. Layman, J. Osborne, and N. Katira. Examining the Compatibility of Student Pair Programmers. In *Agile Conference, 2006*, pages 10–pp. IEEE, 2006.