An Empirical Study of the Effects of Personality on Software Testing

Tanjila Kanij
Swinburne University of Technology
Hawthorn Australia
tkanij@swin.edu.au

Robert Merkel
Monash University
Clayton Australia
robert.merkel@monash.edu

John Grundy
Swinburne University of Technology
Hawthorn Australia
jgrundy@swin.edu.au

Abstract

The effectiveness of testing is a major determinant of software quality. It is believed that individual testers vary in their effectiveness, but so far the factors contributing to this variation have not been well studied. In this study, we examined whether personality traits, as described by the five-factor model, affect performance on a software testing task. ICT students were given a small software testing task at which their effectiveness was assessed using several different criteria, including bug location rate, weighted fault density, and bug report quality. Their personality was assessed using the NEO PI-3 personality questionnaire. We then compared testing performance according to individual and aggregate measures against different five-factor personality traits. Several weak correlations between two of these personality traits, extraversion and conscientiousness, and testing effectiveness were found.

1. Introduction

In our recent survey of professional testers [33], we found that the recipients strongly believed that there are substantial individual differences in tester effectiveness. Given the effort devoted to software testing, and its importance to the quality of software systems, it is natural to investigate the nature of these individual differences. A better understanding of these factors could assist in the identification of individuals with high potential to be good testers, and inform training and development of software testers.

Our respondents indicated that "domain knowledge", "experience", "general intelligence" and "different personal characteristics" contributed significantly to this variance. Some of these characteristics are relatively straightforward, and common across most skilled professions. The "general intelligence" commonly assessed using IQ tests, is positively correlated with job performance in a variety of professions [37], though the basis of this correlation is controversial [32]. Similarly, it is unsurprising that experience and knowledge in the problem domain improve testing performance. Beer and Ramler [6] found that experienced testers had increased domain knowledge that is helpful to understand incomplete and ambiguous specifications. "Different personal characteristics", is however, a less straightforward notion. We believe that these "characteristics" that our respondents referred to, include individual personality traits. Unlike "intelligence" or "domain knowledge", it is not at all clear how particular personality traits might relate to being an effective software tester.

The empirical study reported in this paper was been designed to investigate the effect of personality traits, assessed according to the five factor model of personality [31], on effectiveness of software testing. The study collected the personality profiles of ICT students and measured their effectiveness in a software testing task. We then determined whether there were any relationships between personality traits and effectiveness. A custom software testing task was designed to assess

978-1-4673-5140-9/13 © 2013 IEEE CSEE&T 2013, San Francisco, CA, USA

the effectiveness of students in software testing in our study. Personality profiles were collected using the NEO personality inventory test (NEO PI-3). If specific personality traits have influence on testing effectiveness, these traits could be used to aid recruitment and selection of software testers. This could also help young IT graduates to select the most appropriate IT career for them.

The rest of the paper is organized as follows: Section 2 discusses background and related work of this research study, Section 3 describes the details of this empirical study, Section 4 presents the findings of this study, Section 5 indicates what can threaten the results, Section 6 discusses the details of the findings and finally Section 7 concludes the paper with possible future work.

2. Background and related work

2.1. Personality

While "personality" is a very widely-used concept, a universally-accepted formal definition of the concept remains elusive. Generally, personality is used to refer to enduring non-intellectual personal characteristics that differentiate the individual from others. One standard method of describing a personality is in terms of *traits*, which constitute "... very generalized behaviour patterns in response to emotional tendencies" [31].

There have been many attempts to find a manageable small set of words that represent major personality traits. A result of such attempts is the "big five" factor model [31] which is widely used in personality psychology. The big five factors are extraversion, agreeableness, conscientiousness, neuroticism and openness to experience.

A particularly popular tool for personality assessment research is the Myers Briggs Type Indicator (MBTI), developed based on the theories of Carl Jung [31]. The basic inventory encompasses two kinds of functions and four preference dichotomies - Sensing(S) - Intuition(N), Thinking (T) - Feeling (F), Extraversion (E) - Introversion (I) and Judging (J) - Perceiving (P). The permutation of the four preference dichotomies results in sixteen personality types forming MBTI inventory.

2.2. Research on personality factors related to software testing and debugging

While most software testing research to date has focused on the technical side of the discipline, there have been a few studies that have examined the connection between ability in aspects of testing and various factors relating to personality. Da Cunha and Greathead [18] found that logical and ingenious people, as measured by the MBTI, were good at code review tasks. Almodaimeegh et al. [3] assessed the "locus of control" – that is, the extent to which one believes that an event is the result of external factors than his/her own effort – of programmers. They also investigated the influence of social learning style on debugging skill. They found no significant relation between locus of control and debugging skill, however, individual social learning style and experience were influential on debugging performance. Similarly Shoaib et al. [40], studied the effect of personality traits of students assessed with MBTI, on the effectiveness of exploratory testing and concluded that extravert personality traits are positively correlated with effective exploratory testing. However, whether extraverts will, in general, be good testers still remains an open question.

2.3. Research on other human factors related to software testing

The influence of attributes other than personality, such as experience and work organization, has also been investigated in many works. Beer and Ramler [6] and Itkonen et al. [26] conducted industrial case studies in which they found that more experienced testers used domain knowledge to interpret incomplete specifications, and chose suitable techniques for testing. Organisational issues in testing have also attracted relatively little attention. The limited studies to date [39, 38, 30], however, indicate that attitude and organisational factors are significant, as testing is considered a

boring task and many difficulties relate to organizational factors rather than technical ones. Working within an organization, and the ability to cope with a sometimes very monotonous task, are likely to relate to personality traits.

2.4. Research on personality factors related to programming

There has been a considerable body of research investigating personality types of programmers. Several early studies investigated whether specific MBTI personality traits were over-represented in the programming community, [10, 29]. More recent research examined the impact of these personality types on performance. Cegielski and Hall [14] found that personality type was a predictor of performance in OO programming. However, Darcy and Ma [19] assessed the influence of personality, as measured using the five-factor model, on the performance of student programmers and found no effects.

Other software development tasks have also been examined in this manner. Arockiam et al. [5] investigated the connection between gender, personality traits, and object oriented code comprehension skill. They found that male student participants with cooperativeness and high emotional ability did perform well. Greathead [24] found that introverted people are good at code comprehension. Ahmed et al. [2] found that judging and thinking type students performed better in a software design course.

2.5. Research on personality factors related to software engineering

There has been some research analysing the personality traits of software engineers in general. Capretz [11] administered the MBTI personality assessment test to 100 software engineers ranging from student to professional level. They found that sensing and thinking (ST), thinking and judging (TJ) and intuitive and thinking (NT) types were over-represented in the sample. Sodiya et al. [41] proposed a general test that assesses the five personality factors along with cognitive style with a standard questionnaire and suggests the software engineering role that best suits the assessed personality type. Feldt et al. [21] assessed the personality of 47 professional software engineers, using the International Personality Item Pool (IPIP) inventory, and found that there are different clusters of personalities among them. and each cluster have significant correlation with attitude. Clark et al. [15] assessed the personality of a number of IT students and professionals with Adjective Check List (ACL) and found that both exceptional students and professionals are conscientious, however, exceptional students are introvert while exceptional IT developers are extravert. The difference in personality type between exceptional students and developers raises the possibility that exceptional students might not become exceptional developers.

2.6. Expert views

While there is currently insufficient empirical evidence to make definitive statements on the relationship between personality and testing effectiveness, a number of experts in the area have expressed their opinion on the topic. According to Armour [4], good testers have a “nose for testing” - an intuition that helps them to determine what and how to test. Pettichord [34] listed a number of distinct characteristics of programmers and testers. According to him, testers should tolerate tedium, be skeptical and be comfortable with conflicts. Pol et al. [36] suggests that testers should be creative, accurate and strict in their methodical approach. Black [7] thinks that testers are professional pessimists who possess the curiosity of looking for faults. Weyuker et al. [42] have outlined general and technical skills required for a tester. Capretz and Ahmed [12, 13] collected the job responsibilities mentioned in different job advertisements for different software engineering roles which also includes software testing and connected those responsibilities with required “soft”

skills. Based on their experience and perception, the authors suggested that testers should have attention to detail and should possess good organizational skills, which are common in sensing and judging type people as categorized by MBTI.

The conceptions of these testing professionals suggest that effective software testers have some specific characteristics. However, the empirical evidence supporting links to directly measurable personality characteristics is inconclusive, and in some cases, contradictory.

3. Study design

The goal of this study was to empirically determine relationships between specific personality traits and performance in software testing by software engineering and computer science students, if they exist. This section describes the design of a quasi experiment, in which student participants completed two tasks, an assessment of their personality and an assessment of their effectiveness in software testing.

3.1. Assessment of personality

MBTI has been commonly used in research relating personality and IT practitioners. However, neither the MBTI, nor Jungian personality theories, on which the axes of the MBTI are based, are widely used in modern personality psychology research. A major reason is because the MBTI does not come with norms based on continuous scores, restricting the scope of statistical analysis [9].

We have, instead, chosen to use the NEO personality inventory test (NEO PI-3), proposed by Costa and McCrae [17], and based on the five factor model of personality. In comparison to MBTI, NEO personality inventory tests are widely accepted by personality psychology community, and provide continuous normed scores on the standard five-factor model of personality. The five factors in NEO PI-3 are:

- **Extraversion (E)**: Extraversion is related to sociability, assertiveness, talkativeness and activeness.
- **Agreeableness (A)**: Agreeableness encompasses the expressive quality of admirable human aspects of personality.
- **Conscientiousness (C)**: Conscientiousness is referred to as "Will to achieve" by Digman and Takemoto-Chock [20]. Conscientious people are purposeful, strong-willed and determined.
- **Neuroticism (N)**: This factor covers forms of excessive emotionality. Facets of this include anxiety, angry hostility, depression, self-consciousness, impulsiveness and vulnerability.
- **Openness to Experience (O)**: Openness to Experience is associated with intelligence and intellectual interests.

3.2. Assessment of effectiveness in software testing

3.2.1. Software Testing Task

Unfortunately, there is no standardized method or test for assessment of effectiveness in software testing. In order for us to assess the effectiveness in software testing in this study we have asked students to test a carefully crafted faulty Java program that calculates all possible loop free network topologies given the location of different towers defined with latitude and longitude and find the shortest topology among all. This Java program was originally developed as an assignment as part of an introductory "Programming in Java" unit in Swinburne University of Technology. A number of de-identified student assignments were collected and tested by the researchers. There were 18 different types of bugs encountered in those programs. We chose this program because both the

program and the specifications were of manageable complexity and could thus be understood by participants in a reasonable time.

We selected the least buggy student assignment submission as the basis for the testing task in this study. The program used for our testing task had 1017 lines of code and the maximum cyclomatic complexity of the program was 7. Thirteen bugs found in other student assignments, along with 7 more bugs (which were not found in the student programs) were injected in the assignment program to be used for the study. Thus the testing task program had a total of 20 known bugs in it. The main reason behind injecting detected bugs is that the bugs represent real world bugs as they were created by the student programmers writing the program. Additional bugs were added since the student submissions contained insufficient "obvious" bugs for our purposes. The severity of the injected bugs was decided on the classification taken from [25]. This scheme was chosen because it is widely used and the definition of the categories is clear and straightforward.

To ensure that the injected bugs represent a reasonable approximation of real world bugs, we compared the type of bugs with two standard bug classification scheme, Knuth's errors and Eisenstadt's bug war stories [35]. Table 1 presents the comparison along with the description of the bugs.

Table 1. Description of Injected Bugs and Comparison with Knuth's Errors and Eisenstadt's Bug War Stories

Bug No.	Correct program behaviour	Incorrect program behaviour (Bug)	Fault description	Fault Source	Severity	Knuth's errors	Eisenstadt's bug war stories
1	The output is printed in the console and in the file	The output is printed only on the console	Missing functionality	Students	Low	forgotten function	unsolved
2	Pair of cities are printed with their names	Pairs are printed with numbers	Wrong implementation	Students	Medium	algorithm awry	
3	Prints right number of pairs possible among the given cities	Prints wrong number of pair of cities	Wrong implementation	Researchers	High	blunder or botch	
4	Prints right distance between the pair of cities	Prints wrong distance between pair of cities	Wrong function call	Researchers	High	mismatch between modules	des.logic
5	Prints pairs with right (name/number) of cities	Prints pairs with wrong (name/number) of cities		Researchers	High	blunder or botch	
6	Prints right number of topologies possible among the given cities	Prints wrong number of topologies among the given cities	Wrong index	Students	High	blunder or botch	lex/var
7	Prints right topology distance	Prints wrong topology distance	Wrong index	Students	High	blunder or botch	lex/var
8	Selection of right shortest distance	Selection of wrong shortest distance	Wrong parameter	Students	High	blunder or botch	var
9	Not printing duplicate topologies	Printing duplicate topology		Researchers	Medium	blunder or botch	
10	Right display of distance unit	Wrong display of distance unit	Wrong string literal	Researchers	Medium	trivial typo	
11	Should calculate distance for all values	Cannot calculate topologies for special value (gives exception with 2 cities)		Students	Critical	Forgotten function	lang
12	Calculates right distance for cities with $<$0 degree	Calculates wrong distance for cities with $<$0 degree	Missing Bracket	Students	High	trivial typo	lex
13	Should work with any number of cities	For more than 6 cities gives memory limit exceed exception	Memory	Students	High	Language liability	mem
14	Use of right conditional operator in line 286 wirlessNet.java	Use of wrong conditional operator in line 286 wirlessNet.java	Wrong implementation	Researchers	High	trivial typo	var
15	Use of right conditional operator in line 309 wirlessNet.java	Use of wrong conditional operator in line 309 wirlessNet.java	Wrong implementation	Researchers	High	trivial typo	var
16	Calculates right distance for cities with 0 degree	Calculates wrong distance for cities with 0 degree	Missing equal sign	Students	High	trivial typo	var
17	Should print error message when number of parameters mismatch	No output when number of parameters mismatch	Number of parameter mismatch	Students	Critical	surprising scenario	behav
18	Should print error message for null parameter value	No output for null parameter value	Null parameter	Students	Critical	surprising scenario	behav
19	Should print error message when parameter types mismatch	No output with parameter type mismatch	Parameter type mismatch	Students	Critical	surprising scenario	behav
20	Right method name "getTopologies"	Wrong method name "getTopology"	Wrong implementation	Students	Low	trivial typo	

3.2.2. Assessment metrics

There are a number of proposed metrics in the literature for assessing testing effectiveness [22, 28]. Three metrics were used for our purpose: Bug location rate, Weighted fault density, Bug report quality. Given that there are no universally-accepted criteria for tester effectiveness, we judged that these three metrics were plausible and practical for use in our experiment.

- **Bug Location Rate:** This is measured as the number of bugs found in comparison to the time taken for testing [25].

Bug Location Rate $= \frac{Numberofbugsfound}{Timetakenfortesting(inminutes)}$

Bug location rate could range from 0 to infinite.

- **Weighted Fault Density:** The weighted fault density can be measured as:

  Weighted Fault Density $= \frac{(W_1 * S_1)+(W_2 * S_2)+(W_3 * S_3)+(W_4 * S_4)}{N}$

  where:
  W denotes assigned weights $W_1 > W_2 > W_3 > W_4$. S denotes severity of faults assigned according to [25]. N denotes number of total bugs found.

  There were 4 critical, 11 high, 3 medium and 2 low bugs in the software. For this experiment we used the following weights: $W_1 = 0.4, W_2 = 0.3, W_3 = 0.2$ and $W_4 = 0.1$. The total weighted fault density could range from 0 to 0.4.

- **Bug report quality:** This is assessed using the IEEE standard of Test Documentation [1]. According to the standard, there are 8 separate documents that should be given as part of a bug report. As our test sessions were small, as well as the provided software to test was also a small and simple one, we did not expect the participants to provide separate document(s) on each of the bugs found. We examined whether any information on each of the 8 standard deliverable fields were present for each of the bugs reported in the bug report prepared by a student and assigned a score from 0 to 1 for each of the field. Thus the total score of bug report quality could range from 0 to 8.

The total number of bugs found was calculated as the number of reported valid bugs (Total number of reported bug - total number of reported invalid/false bug).

3.2.3. Assessment of overall effectiveness

Unfortunately in the testing literature there is no indication of how to assess overall effectiveness using the above defined metrics. For assessment of overall effectiveness using these metrics we followed two possible approaches:

- **Total score:** In this approach we summed the score of bug location rate, weighted fault density and bug report quality of a participant to derive overall score. The score is denoted as O_{sum}.

  O_{sum} = bug location rate + weighted fault density + bug report quality

- **Weighted total score**: In this approach we assigned different weights to the assessment metrics according to their importance on total score to get an overall score. According to Kaner [27] the number of bugs found by a tester may be affected by a variety of factors. He emphasized qualitative assessment of bug report quality than accounting for bug counts [28]. In our previous study [33], we found that 'Quality of bug report" is more important than "Severity of bugs found". Therefore, we assign more importance to bug report quality. The other two proposed metrics are based on bug count and we can not distinguish the varying importance of those. We assign the following weights to the three proposed metrics (bug location rate = 0.3, weighted fault density = 0.3 and bug report quality = 0.4) and sum the weighted metrics to get weighted total score, denoted as O_{wsum}.

  O_{wsum} = (0.3 * bug location rate) + (0.3 * weighted fault density) + (0.4 * bug report quality)

3.2.4. Validation of the instrument

After a pilot study conducted with 9 volunteer student participants, minor changes were made in the description of the specification document provided with the testing task.

3.3. Participants

The participants in this study were Computer Science and Software Engineering students of the Faculty of Information and Communication Technologies at Swinburne University of Technology and the Faculty of Information Technology at Monash University. Both universities are located in Melbourne, Australia. The researchers attended lectures for ICT undergraduate and postgraduate coursework units at both universities to invite students to participate.

3.4. Experimental procedure

After each of the participants signed a consent statement, they were assigned a unique number and were provided with a small demographic questionnaire on whether they had any professional experience in software testing, any experience using testing tools, any professional experience in other IT areas, and whether they had completed any university units on software testing. Once the demographic questions were completed the participants were presented with the self scoring NEO PI-3 form. After the completion of the personality assessment form the participants were given the specification document of the testing task program. They were asked to verify if the program meets the specification and report any deviations as bugs. The debugging environment used was Eclipse (version 3.5.0) which is a standard IDE used by both universities. No testing technique was specified to the participants. At the end of the test each participant received $25 for participation. The participants were provided with their personality profile form named "Your NEO Summary". The data was collected over the period September 2010 to October 2011.

4. Results

4.1. Participant demographics

A total of 48 students from 18 to 35 years of age, participated in the research study. The majority (69%) of participants were male. 22.3% participants had professional experience in software testing. 31.3% had studied a specialized unit on software testing. 27.1% had experience of using testing tools and 37.5% had experience in other areas of IT.

4.2. Population distribution

The scales of NEO personality inventory tests measure factors that follow an approximately normal distribution in the broader population. We have applied Shapiro-Wilk Test [8] and the results indicate that our population distributions do not differ significantly from normality.

4.3. Scores on personality traits and effectiveness in testing

Table 2 shows the minimum, maximum, average and standard deviation of the T-scores on five personality factors and the scores on the effectiveness measurement metrics obtained by our participants. The T scores are calculated based on the norms provided with the NEO PI-3 personality assessment test.

Table 2. Distribution of scores (N = 48)

	Minimum	Maximum	Average	Standard deviation
Neuroticism	32	74	54.94	10.38
Extraversion	20	70	50.42	9.32
Openness to experience	35	80	54.36	10.02
Agreeableness	29	74	48.27	9.62
Conscientiousness	27	66	47.25	8.62
O_{sum}	0.63	3.87	1.91	0.85
O_{wsum}	0.24	1.51	0.73	0.34
Bug Location Rate	0.02	0.37	0.12	0.063
Weighted Fault Density	0.1	0.33	0.23	0.07
Bug Report Quality	0.5	3.5	1.56	0.83

4.4. Correlation between personality traits and effectiveness in testing

Table 3 shows Pearson correlations between the five personality factors measured, and our testing effectiveness metrics. We found, at significance level $p < 0.05$, there is a negative association between extraversion and O_{sum}, O_{wsum} and bug report quality. We also see a positive, though weak association between conscientiousness and bug location rate and a weak negative association between conscientiousness and weighted fault density.

Table 3. Correlations (N = 48)

	Neuroticism	Extraversion	Openness to experience	Agreeableness	Conscientiousness	O_{sum}	O_{wsum}	Bug Location Rate	Weighted Fault Density	Bug Report Quality
Neuroticism	1	-.329*	-0.136	-0.135	-.457**	0.034	0.036	-0.122	0.02	0.043
Extraversion		1	.401**	-0.231	.375**	-.267*	-.267*	0.038	-0.133	-.265*
Openness to experience			1	-0.235	0.177	0.161	0.165	-0.025	-0.154	0.179
Agreeableness				1	-0.121	0.167	0.173	-0.034	-0.215	0.191
Conscientiousness					1	0.026	0.026	.251*	-.241*	0.028
O_s						1	1.000**	.258*	0.085	.996**
O_{ws}							1	.248*	0.071	.998**
Bug Location Rate								1	-.310*	0.214
Weighted Fault Density									1	0.028
Bug Report Quality										1

*. Correlation is significant at the 0.05 level (1-tailed)
**. Correlation is significant at the 0.01 level (1-tailed)

5. Threats to validity

This study was conducted on undergraduate students, most with no professional experience in testing, and with a limited number of participants, at two universities in Victoria, Australia. Replication with different, preferably larger, cohorts would be required to have confidence that our results are generally applicable.

Participants were paid $25 for their participation in the study. This reward might have been their primary motivation for participating and thus distorted their efforts. However, the observed enthusiastic participation of the participants suggests that this was not in fact the case.

One possible threat might be the reliability of our software testing effectiveness test. The bugs injected in the instrument do represent realistic bugs. However, the very small scale of the task is not representative of testing large, real-life software systems. Another threat to the construct validity is that the results may be dependent the domain and programming style used in the software forming the basis of the testing task.

6. Discussion

The purpose of this study was to investigate the effect of different personality traits, as captured by the five-factor model, on effectiveness in software testing, as assessed with a custom software testing task. We recruited nearly 50 students to undertake a detailed NEO test for personality traits and to undertake a software testing task. To find any association we closely examined the Pearson correlations of the five factor personality traits with the overall effectiveness in testing. We also combined several different measures of testing effectiveness in different ways. In contrast to our initial assumption and the anecdotal opinions of testing experts, we did not find strong correlation between the personality factors and the testing effectiveness measures.

A weak negative correlation, however, is seen between extraversion and overall effectiveness, as discussed in Section 4.4. This result differs from the study of Shoaib et al. [40] that concluded extravert people were good at exploratory testing. A negative correlation also exists between extraversion and bug location rate. As suggested by the type descriptions [16], extravert people are assertive, active and talkative. As such it is rather surprising that people who are enthusiastic about communication may not be as good as others at reporting bugs.

Another personality trait, conscientiousness, showed a weak positive correlation with bug location rate and a weak negative correlation with weighted fault density (Section 4.4). Bug location

rate takes time spent in testing in to account with the total number of bugs found in testing. According to [16] conscientious people are determined, punctual and reliable that complements the positive correlation between bug location and conscientiousness. On the other hand, weighted fault density is more concerned with the severity of found bugs. The negative correlation of conscientiousness with weighted fault density rises the question whether highly conscientious individuals do not pay as much attention to the severity of bugs? The above two findings indicate that highly conscientious individuals might be more interested towards finding *more* bugs without giving as much attention to the *severity* of the bugs found.

Conscientiousness is closely linked to the Judging-Perceiving dimension of MBTI [23]. According to Capretz and Ahmed [12, 13], sensing and judging individuals categorized by MBTI would be good at testing. Our finding that conscientiousness has influence on testing effectiveness, thus, partially supports Capretz and Ahmed's perception. However, the exact effect of conscientiousness should be examined by more specific experiments.

7. Conclusions

In this empirical study we attempted to identify if correlations exist between the five major personality traits, as measured by the NEO inventory, and student performance on a representative software testing task, as measured by three performance metrics. We carried out a study with nearly 50 Computer Science and Software Engineering students, assessing their personality traits and their performance on our testing tasks. Our results indicated weak associations between both extraversion and conscientiousness, and effectiveness in testing. We plan to replicate the experiment with a larger number of participants to test the validity of this finding. We plan to replicate our study with different kinds of testing tasks. Ideally we would also like to replicate our study with actual testing professionals. More precise effects of the extraversion and conscientiousness traits should be examined with a more focused experiment in which the influence of the other independent variables can be kept to a minimum.

References

[1] IEEE Standard for Software Test Documentation. *IEEE Std 829-1998*, page i, 1998.

[2] F. Ahmed, P. Campbell, A. Jaffar, S. Alkobaisi, and J. Campbell. Learning & personality types: A case study of a software design course. *The Journal of Information Technology Education*, 9, 2010.

[3] H. Almodaimeegh and J. Harrold. Predicting Debugging Successs: An Investigation of the Relationship between Learning Styles, Personality Traits, and Computer Program Debugging . In G. Siemens and C. Fulford, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2009*, pages 2702–2710, Honolulu, HI, USA, June 2009. AACE.

[4] P. G. Armour. The Unconscious Art of Software Testing. *Communications of the ACM*, 48(1):15–18, 2005.

[5] L. Arockiam, T. L. A. Beena, K. Uma, and H. Leena. Object-Oriented Program Comprehension and Personality Traits. In *Proceedings of SMEF*, 2005.

[6] A. Beer and R. Ramler. The Role of Experience in Software Testing Practice. In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications*, pages 258–265, Washington, DC, USA, 2008. IEEE Computer Society.

[7] R. Black. Being a 'good' tester: Attitudes, skills, and growth. http://www.rbcs-us.com/documents/BeingaGoodTester.pdf.

[8] S. Boslaugh and P. A. Watters. *Statistics in a Nutshell*. O'Reilly, Sebastopol, CA, USA, 2008.

[9] G. J. Boyle. Myers-Briggs Type Indicator (MBTI): Some Psychometric Limitations. *Australian Journal of Psychology*, 30, March 1995.

[10] C. Bush and L. Schkade. In search of the perfect programmer. *Datamation*, 31(6):128–132, 1985.

[11] L. F. Capretz. Personality Types in Software Engineering. *Int. J. Hum.-Comput. Stud.*, 58(2):207–214, 2003.

[12] L. F. Capretz and F. Ahmed. Making Sense of Software Development and Personality Types. *IT Professional*, 12:6–13, 2010.

[13] L. F. Capretz and F. Ahmed. Why Do We Need Personality Diversity in Software Engineering? *Journal of Information Systems Education*, 35:1–11, March 2010.

[14] C. G. Cegielski and D. J. Hall. What Makes a Good Programmer? *Communications of the ACM*, 49:73–75, October 2006.
[15] J. G. Clark, D. B. Walz, and J. L. Wynekoop. Identifying Exceptional Application Software Developers: A Comparison of Students and Professionals. *Communications of the Association for Information Systems*, 11, 2003.
[16] J. Costa, P. T. and R. R. McCrae. The NEO Personality Inventory Manual. *Psychological Assessment Resources.*, 1985.
[17] P. T. Costa and R. R. McCrae. *Revised NEO Personality Inventory (NEO-PI R) and Neo Five Factor Inventory (NEO-FFI)*. Psychological Assessment Inventories, 1992.
[18] A. D. D. Cunha and D. Greathead. Does Personality Matter? An Analysis of Code-Review Ability. *Communications of the ACM*, 50(5):109–112, May 2007.
[19] D. P. Darcy and M. J. Ma. Exploring Individual Characteristics and Programming Performance: Implications for Programmer Selection. *Hawaii International Conference on System Sciences*, 9:314a, 2005.
[20] J. M. Digman and N. K. Takemoto-Chock. Factors in the Natural Language of Personality: Reanalysis, Comparison, and Interpretation of Six Major Studies. *Multivariate Behavioral Research*, 16(2):149–170, 1981.
[21] R. Feldt, L. Angelis, R. Torkar, and M. Samuelsson. Links between The Personalities, Views and Attitudes of Software Engineers. *Inf. Softw. Technol.*, 52:611–624, June 2010.
[22] N. Fenton and S. L. Pfleeger. *Software Metrics (2nd ed.): A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 1997.
[23] A. Furnham. The Big Five Versus the Big Four: The Relationship Between the Myers-Briggs Type Indicator (MBTI) And NEO-PI Five Factor Model of Personality. *Personality and Individual Differences*, 21(2):303 – 307, 1996.
[24] D. Greathead. MBTI Personality Type and Student Code Comprehension Skill. Technical report, www.ppig.org, 2008.
[25] M. L. Hutcheson. *Software Testing Fundamentals: Methods and Metrics*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.
[26] J. Itkonen, M. V. Mantyla, and C. Lassenius. How Do Testers Do It? An Exploratory Study on Manual Testing Practices. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 494–497, Washington, DC, USA, 2009. IEEE Computer Society.
[27] C. Kaner. Don't Use Bug Counts to Measure Testers. *Software Testing & Quality Engineering*, page 80, May/June 1999.
[28] C. Kaner. Measuring the Effectiveness of Software Testers. *Software Testing Analysis & Review Conference (STAR) East*, May 2003.
[29] M. L. Lyons. The DP Psyche. *Datamation*, 31(16):103–105, 1985.
[30] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville. 'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 602–611, Washington, DC, USA, 2007. IEEE Computer Society.
[31] D. P. McAdams. *The Person: An Integrated Introduction to Personality Psychology*. Harcourt, Inc, Orlando, FL, USA, 3 edition, 2001.
[32] D. C. McClelland. Intelligence Is Not the Best Predictor of Job Performance. *Current Directions in Psychological Science*, 2(1):5–6, February 1993.
[33] R. Merkel and T. Kanij. Does the Individual Matter in Software Testing? http://www.swinburne.edu.au/ict/research/sat/technicalReports/TC2010-001.pdf.
[34] B. Pettichord. Testers and Developers Think Differently: Understanding and Utilizing the Diverse Traits of Key Players on Your Team. *Testing & Quality*, 2, January-February 2000.
[35] J. Ploski, M. Rohr, P. Schwenkenberg, and W. Hasselbring. Research Issues in Software Fault Categorization. *SIGSOFT Softw. Eng. Notes*, 32, November 2007.
[36] M. Pol, R. Teunissen, and E. V. Veenendaal. *Software Testing:A Guide to the TMap Approach*. Addison-Wesley (E), 2001.
[37] M. J. Ree and J. A. Earles. Intelligence is the Best Predictor of Job Performance. *Current Directions in Psychological Science*, 1(3):86–89, June 1992.
[38] J. Rooksby, M. Rouncefield, and I. Sommerville. Testing in the Wild: The Social and Organisational Dimensions of Real World Practice. *Comput. Supported Coop. Work*, 18:559–580, December 2009.
[39] H. Shah and M. J. Harrold. Studying Human and Social Aspects of Testing in a Service-Based Software Company: Case Study. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '10, pages 102–108, New York, NY, USA, 2010. ACM.
[40] L. Shoaib, A. Nadeem, and A. Akbar. An Empirical Evaluation of The Influence of Human Personality on Exploratory Software Testing. In *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International*, pages 1–6, December 2009.
[41] A. S. Sodiya, H. Longe, S. Onashoga, O. Awodele, and L. Omotosho. An Improved Assessment of Personality Traits in Software Engineering. *Interdisciplinary Journal of Information, Knowledge and Management*, January 2007.
[42] E. J. Weyuker, T. J. Ostrand, J. Brophy, and R. Prasad. Clearing a Career Path for Software Testers. *IEEE Software*, 17:76–82, March 2000.