

School of Computing Science,
University of Newcastle upon Tyne



Code review and personality: is performance linked to MBTI type?

Alessandra Devito Da Cunha and David Greathead

Technical Report Series

CS-TR-837

April 2004

Copyright©2004 University of Newcastle upon Tyne
Published by the University of Newcastle upon Tyne,
School of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK.

Code review and personality: is performance linked to MBTI type?

Alessandra Devito Da Cunha, David Greathead

¹ Centre for Software Reliability
University of Newcastle upon Tyne
Claremont Tower
Newcastle upon Tyne NE1 7RU
United Kingdom

Abstract

Dependability frequently focuses on approaches such as rigorous design and fault tolerance. A final software product is strongly influenced by the people who created it. Large variations in individual performance in software development have been observed. Some form of innate programming ability might account for some of the differences. The influence of personality on computer programming has been examined but typically treated programming as a single process. As Weinberg [1] suggested, different aspects of software development require different abilities so there may be no one personality type which is better overall. To examine this, a study was carried out to discover if there was a particular personality type which tended to perform better at one specific task, in this case code review. It was found that people with a specific personality type performed significantly better on the task.

Keywords: Personality, Code Review, Testing, HCI

Introduction

The aim of the current study was to investigate if there is a specific personality type whose members perform better at code-review. This research was partly inspired by theorists [2], [3], [1] who pointed out that the programming process encompasses various aspects which require different abilities. In addition it was noticed that there were large variations in individual performance and some reasoned that this variation may be due to differences in personality type. To examine this, a study using university second year undergraduate students was

carried out including an analysis of their performance in a code-review task and an assessment of their personality types.

Phases of programming

Software is developed by progressing through various phases which ultimately lead to the final product. It is important to emphasise here that each phase of the programming process involves different facets of its production, i.e. each phase will need a different approach from the individual undertaking the task.

To illustrate the phases involved in software production, an idealised model (the Waterfall Model) was taken [4]. This model suggests that the next phase starts only when the previous phase is finished. The authors accept that this is not always strictly true; however the model is useful for clarity and ease of explanation, provided that this caveat is remembered.

To begin with, when new software is going to be developed a study called a feasibility study is carried out. An analysis of costs and benefits will determine whether or not it is viable to carry on with the production. If it is decided that it is feasible to construct the new software, it is necessary to establish its requirements. During this phase, all the functions and descriptions of the new product should be specified.

With these specifications in hand, the next phase starts: during the design phase, details about how to meet the specifications (established in the previous phase), are created. The tools and the techniques to solve the problems are also established here.

As soon as the design is concluded, coding takes place. For this it is necessary to adopt a programming language and code the implementations of the specifications and design. If the software is a large and complex system, it may be divided into chunks which will later be integrated. It is possible that while integrating these chunks, bugs are added to the code.

It is also possible that, even in a small piece of software, bugs are inserted into the code. These bugs will prevent the program from running as expected, so there is a need to find and remove them. The next phase is to test the code to uncover any bugs in the program (and later remove them).

Testing consists of running the program and looking for unexpected behaviour. It is one way to discover if there are bugs in the program. Reviewing the code in detail will also highlight possible bugs which can then be removed. However the process of debugging code often creates further bugs so it is necessary to test and debug the program more than once.

Documentation is the next phase, although some theorists suggest that this phase is done between specification and design. Documenting consists of registering the specifications for the user manual as well as technical information for reference.

At the conclusion of all of these phases, the software is ready to be released. However it is nearly always the case that the product will undergo maintenance. This can occur for two reasons: because the product presents some failure, or because it requires updating. Again there is a need to test and debug the program after maintenance changes.

Code-review

For the purposes of this research, code review is the term adopted to refer to the act of finding the bugs present in the software (i.e., it is not concerned with correcting them). This practise can be seen during testing, where the “code-reviewer” analyses the code without executing it and points out possible mistakes. The “code-reviewer” is generally a person who has no other involvement in the development process.

As the bugs are only exposed by the reviewer, the act of correcting them is a task for the debugging team. Although it is possible that the reviewer identifies a “false bug”, there is a

reduced danger of “false bugs” leading to new errors as the author(s) is present to justify their work. Reviewing the code should occur frequently during software development as it is possible that new bugs are added to it during each phase.

Psychology and Programming

Cooperation between these two sciences was proposed by Weinberg in 1971 [1], and since then many other theorists have tried to look at some of the psychological aspects involved in the programming process.

In his book, Weinberg [1] presented evidence supporting his hypotheses regarding psychology and the programming process and, while this information tended to be anecdotal in nature, it was nonetheless compelling. He covered such diverse areas as personality, intelligence, motivation, training, experience, as well as considering what make a good program and the various aspects of programming in groups or teams.

Weinberg, in fact, presented ideas that could guide (directly or indirectly) authors to conduct research on human-computer-interaction, variation in performance and personality styles.

Variation in Performance

Some theorists noted that there were surprisingly large variations in individual productivity and accuracy while executing parts of the software development process. Boehm [5] described the variation as being between a factor of 10 and a factor of 30.

Pressman [2] noticed that programmers with the same background performed differently at the same task (debugging a program). He said that there may be some “innate human-trait” behind such variation, as there are some programmers who are good at debugging while others are not.

Weinberg [1] suggested that this variation in productivity may be related to differences in the type of task. He explained that as each phase of the programming process requires different actions, each action will then require some specific individual skills. In this case it can be assumed that a programmer will perform better at a certain phase of the programming process and not as well at the others phases. Furthermore Weinberg said that it is possible that the performance in the task is a result of the programmers' individual characteristics.

There have been a number of studies carried out based on individual characteristics influencing performance at work [6-11]. Individual characteristics can be assessed using some of the many personality tests. There are some personality tests which are biased towards a clinical sample and are aimed at aiding diagnosis of disorders (such as the Minnesota Multiphasic Personality Inventory, or MMPI) while others tests are not aimed at diagnosis (such as the MBTI and 16PF).

Many of the studies carried out within personality and computing science were done with the Myers Briggs Type Indicator (MBTI). This is the primary reason why this personality assessment was chosen in the current study.

Bishop-Clark [12] analysed the personality types of college students in a programming class and made some suggestions about the right type for each phase of the programming process due to the fact that, according to Weinberg [1] each phase is distinct, requiring different types of people to work on them. Recently, Capretz [13] analysed the personality types of software engineers. He found that while the MBTI describes 16 personality types, 24% of the engineers were all of the same type (ISTJ) which characterises the person as quiet, serious, concentrated, logical and realistic. A person presenting this type is technically oriented, does not like dealing with people and when working prefers to deal with facts and reasons. However, there is no indication whether a programmer with such characteristics performs better than programmers with other personality characteristics. Other personality types were

given, but again there was no evidence which could reinforce the idea that such types would be better at certain phase of programming than others.

MBTI

The Myers Briggs Type Indicator (MBTI) is one of the most widely used personality assessments [14-17]. It is based on Jung's theory of psychological types relating to three bipolar factors or dimensions: extroversion/introversion (EI), sensing/intuition (SN) and thinking/feeling (TF). When the MBTI was developed, Myers and Briggs added another pair of characteristics which related to judgment and perception (JP).

With regard to personality and career choice in general the sensing/intuition and thinking/feeling dimensions are responsible for attraction to certain jobs while extroversion/introversion and judgement/perception determine the attitude people maintain.

Myers [18] explained that the EI dimension is concerned with the way people tend to "recharge their energy". Extroverts will focus their attention on other people through the external environment, while introverts will be fully recharged after staying with close friends or family (or by being alone) in an internal environment. This then has an influence on career choice in that this dimension influences people to choose the sort of occupation related to their style: extroverts need contact with people while introverts get stressed when they have too much contact with people as they would prefer to work with impressions and ideas. However interest in jobs is mainly determined by the dimensions SN and TF.

The sensing/intuition dimension is related to how people acquire information, i.e. whether it occurs through the five senses in a concrete manner or through intuition using imagination and inspiration. The third dimension is that of thinking/feeling and it is concerned with how people make decisions. As the name suggests, the decisions can be made in two ways – either

following some logical sequence of facts or making decisions based on a people-centred opinion with more of an emphasis on feelings rather than logic.

These two dimensions of preferences are responsible for the cognitive dimensions which determine how people feel attracted to and satisfied by their career choice. Both of these factors are also important tools in solving problems as they help in the improvement of problem solving skills. It is important to emphasise here that one type is not regarded as being better than another - people simply need to be aware of how to take advantage of their type and then work with their preferences and their abilities.

A person's lifestyle is determined by the judging/perceiving dimension. If people are controlled, orderly, and plan everything carefully then they can be said to have a judgement orientation. On the other hand, if they are more flexible, spontaneous and adaptable to new situations their lifestyle is more orientated towards perception. In a person's career, judgement and perception influence how the job is performed.

The MBTI is intended to be used with "normal individuals in counselling and within organizations" [17]. Smither [15] explains that such an instrument is not a useful tool in recruiting people. In fact the MBTI is a useful tool in relocating people in organizations, i.e. selecting current employees for a special task and to improve work communication/interaction. As such it is a popular personality measure within industry, being familiar to both employers and employees alike.

In the second edition of his book, Weinberg [19] states that he would have written a completely different chapter about personality if he had known the advantages of the MBTI. He adds that this assessment is dealing "with normal personality differences" (19, p.8i) as some personality tests are related to mental disorders (such as the MMPI), i.e. they do not see the person as being normal.

Edwards, et al [14] state that despite the popularity of MBTI it has been criticized on a number of levels. One aspect is that some theorists consider typologies as a reductionist approach to reproduce diversities among people. Also “there is evidence that the MBTI dimensions, rather than clustering into distinct categories or interacting with each other, are more akin to four additive main effects” [14].

Methodology

Apparatus

In order to collect the relevant information in this study, two different instruments were employed, these being the Myers-Briggs Type Indicator (MBTI) and a code-review task.

The MBTI Step 1, European English Edition was used to assess participants’ personality. This is an 88 item forced response personality inventory, which returns scores on four bipolar scales. Four letters are returned to indicate the type of preference, for example an individual would have either I or E as one of their letters (for Introversion or Extroversion), with a corresponding value indicating the strength of this preference. Similar scores are obtained for Sensing (S) or Intuition (N); Thinking (T) or Feeling (F); and Judging (J) or Perceiving (P).

The code-review task consisted of four pages (282 lines) of Java code, which had been written by an experienced programmer. The program was a pattern search program which would operate on an ASCII file. After being examined for the presence of bugs, 16 semantic bugs were inserted into the code by the programmer, which varied in difficulty (easy, medium or hard) as rated by the programmer. The program was accompanied by a two page manual and API, including an example of the output the code produced when executed, and the ASCII file which was used for the example. A title page provided instructions on how to complete the task. One of the participants returned the task essentially blank and as such was excluded from certain statistical tests.

Java was chosen as the language as it was the only language that all of the participants had knowledge of; it being the language which was taught during the first year at the university.

Participants

Sixty-four participants completed both stages of the study. The participants were all undergraduate students from Newcastle University in the United Kingdom. While participants were unselected for age and sex, the majority (81%) were male and 77% were aged 19-21 due to the nature of the population. Participants were all awarded extra marks in their course for each aspect of the research in which they took part. Three prizes per programme of study were also awarded (of £10, £20 and £30) for the participants who scored more highly on the code review task. This was in order to encourage participants to apply themselves to the task.

Procedure

The two instruments were administered on separate occasions, in regularly scheduled, one-hour lecture slots. The code review task also occupied part of the adjacent lecture slot. At both stages of the research, participants were reassured that their data would not be used in such a way that they could be individually identified.

The code review task was administered in January 2003. For this task, participants were given one and a half hours. Participants were informed that this was an individual task and were asked not to talk to one another. They were also to spread themselves out as much as possible in the lecture theatre. As well as being reminded that they would receive extra course credit for their participation, the participants were informed that there would be three prizes for each of the programmes of study, awarded to the highest scoring participants in each programme. They were also given additional information about the task in that all of the errors in the code were semantic errors, and that they did not have to correct the errors, only identify them in some way. They were not informed how many errors there were in total.

The MBTI was administered in March 2003. Participants were allowed up to one hour to complete the questionnaire and were free to leave the lecture theatre once they had finished. Participants were also offered an individual feedback session if they desired. The researchers were available to answer any questions participants may have had with regard to the questionnaire.

Results

For the purpose of analysis, the various bugs were weighted according to their difficulty. Given that the experienced programmer may have had a slightly different concept of what constituted an easy or a difficult bug to locate as compared with the novice programmer, it was decided not to use the experienced programmer’s weighting of the bugs. Instead, the difficulty of the bug, and therefore the weight awarded for statistical analysis was based on the number of participants who found that particular bug. For example, if a bug was found by almost all participants, then it was classed as ‘easy’ and given a weight of two points for analysis, whereas if almost no participants found a bug, it was weighted as ‘difficult’ and given a weight of four. Bugs in between were ‘medium’ and weighted three. Ultimately, there were five easy, four medium and seven difficult bugs. Using this method to weight bugs produced a markedly different result, indicating that the students’ opinion regarding what constituted an easy, medium or difficult bugs differed from that of the experienced programmer.

In order to examine the possible links with MBTI type and code review ability (score), a number of correlations were carried out. The results are presented in Table 1 below.

Table 1 - Correlation matrix between MBTI type and code review score

	Extroversion	Sensing	Thinking	Judging
Code Review – Pearson correlation	-0.197	-0.251	0.197	0.000
Sig (2-tailed)	0.121	0.047	0.122	0.998

As can be seen, the only significant correlation was that between the Sensing scale and the code review score. High scores on the Sensing scale represent an individual with a high *sensing* preference, while low scores represent an individual with a high *intuitive* preference. As this is a negative correlation, it indicates that people who are more *intuitively* inclined performed significantly better on the code review task than *sensing* types. There were minor, non-significant correlations with the *extroversion* and *thinking* scales but no correlation whatsoever with the *judging* scale.

Further examination of the data revealed some interesting information. If two of the letter types were considered together, scores could be considered on a four-group basis. While doing this, the most marked differences came with the SN and TF scales. Examining the mean code review scores shows that the NT students scored 9.10 as compared to the non-NTs who scored 6.14 on average. This illustrates that, with this sample and according to letter categories only, the NT individuals were better able to perform the code review task than the non-NT people. The mean scores for these four types are included in Table 2.

Table 2 - Mean code review score by SN/TF types.

	F	T
N	8.71	9.10
S	4.27	6.62

As can be seen from the table, the most marked difference was between NT participants and SF participants, who achieved on average less than half the score of the NT participants. A t-test comparing NTs with non-NTs yielded a significant result, illustrating that NTs were better at the task than non-NTs (1-tail sig = 0.039, $t = 1.801$, $df = 61$).

Following this, a regression analysis was carried out using the continuous data obtained from the IE, SN and TF scales. The results of this are summarised below.

Table 3 - Regression analysis summary

R Square	0.016	
Anova Sig	0.016	
	t	Sig
Constant	3.295	0.002
Sensing	-2.363	0.021
Thinking	1.358	0.180
Extroversion	-2.218	0.030

As Table 3 shows, this regression model is a significant one and can account for 16% of the variance of the score on the code review task with personality alone, with the *Sensing/Intuition* scale being particularly relevant.

In addition to scores obtained on the code review task, the first year programming module grades were also obtained for most of the participants. A t-test was carried out to compare the grades of NT participants with non-NT participants on these two programming modules. Results of these tests are summarised in Table 4 below.

Table 4 - T-tests between programming grade and NT/non-NT

		N	Mean	t	df	Sig (1-tail)
CSC131	NT	16	69.69	1.408	43	0.083
	non-NT	29	62.62			
CSC132	NT	15	68.80	2.117	42	0.020
	non-NT	29	58.38			

As can be seen, there was a significant difference between NTs and non-NTs for CSC132 (Algorithms and Data Structures in Java), but not for CSC131 (Introduction to Programming and Software Engineering in Java). Examining the means shows that the mean difference between NTs and non-NTs was approximately ten percent, with the NTs scoring more highly. There was also a difference of approximately seven percent for CSC131, with NTs scoring more highly, however this latter difference was not statistically significant.

Discussion

The characteristic of NTs ([12], page35) is that they perceive the world through their intuition, i.e. they gain their insights by meanings rather than by observing facts. They also make their

judgements by thinking, i.e. by logical connections of the facts rather than by weighing the decisions. They are always “looking at the possibilities, theoretical relationships, and abstract patterns” ([12], page35). The way in which they make their judgements is impersonal. NTs are known as “logical and ingenious” and they are best at solving problems (the task of finding bugs in programs could also be considered a problem solving task) within their field of special interest (in this case, programming).

Although there is no evidence about how well or otherwise the job is performed, there is a list of “occupational choices for NT” (appendix D, [12], p259). Some of these were computer related jobs however only a little detail other than this was presented there. In a sample of 86 computer systems analysts, 41.86% were NTs; in another sample of 57 computer specialists, 36.98% were NTs; and the NTs represented 36.50% of a sample of 200 computer programmers.

The statistics above suggest that NTs do feel a certain attraction to computer related careers. This may indicate that there are NTs who are unaware of their potential as good code-reviewers. If a company organises its employees according to their personality types and their potential abilities, productivity and quality may be improved [20].

As can be observed from the results, it is not merely the case that in this sample the NTs performed better at the task than the other letter types, but that the difference was so marked between NTs and their opposite types, SFs, who scored less than half as well as the NTs on this task. It is not the authors’ intent to suggest that SFs be precluded from performing code review tasks. Indeed, the percentage of variance accounted for in the code review score by MBTI type was relatively modest. However, it is worth remembering that this difference could be accounted for in the way NTs look at the world. If this is the case, then it would be advantageous to encourage code reviewers in general to be aware of the way NTs think as this may aid them in code review [20]. As Briggs Myers & McCaulley [12] state, people can

generally act in a way which is not congruent with their type, even if it would not be their first choice in the everyday world.

Future research

It can be assumed that as NTs present particular qualities which contributed to the success in the code review task, there are some types which may be linked to success at the other steps of the programming process. As Weinberg indicates, each part of the programming process has a specific task to be completed which requires specific “combinations of skills and personality traits” [1]. Thus it would be useful to look at the other stages of the programming process and the personality types involved. This is suggested as one particular avenue of future research, and work to this end has already been begun with regard to design and coding.

It could also be true that the reason the NTs performed better than the other types on this task, was something other than the fact that it was a code review task. One possibility is that there was something in the nature of the program itself, or the type of bugs presented which aided the NTs in the task. For instance, all of the bugs were semantic in nature, rather than syntactic, which may have aided the NTs due to the way they approached the task. In order to examine this possibility, the authors are in the process of replicating the study, substituting the program used with another one, of a different type, written by a different programmer.

It is also worth remembering that in this sample, the NT students performed significantly better than the non-NT students on one of their level-one programming modules, but not the other. Given that the participants performed better on the Algorithms and Data Structures in Java module rather than the general introductory module suggests that the NT participants may indeed be predisposed to high performance on some types of tasks, and not on all aspects of the software development process. This therefore is further reason to carry out more research into MBTI type and performance at various aspects of software development.

It would also be advantageous to examine performance on code review, and other tasks when considering the full, four-letter types (e.g. INTJ). This was not statistically practical with the current study due to the sample size of 64. While at least one of each MBTI type was obtained within this sample, some of the types had only one or two participants, thus making any statistical analysis inappropriate. In order, therefore, to examine the full, four-letter types, further research would have to be carried out to increase the sample size. Despite the low sample size for this type of analysis, it is interesting to note how the results differ from those discussed by Capretz [13] in that, while Capretz reported 24% of participants as being ISTJ, the current study only reported 6% as ISTJ (four participants). The largest number of participants of any one type was for the ENTP type, there being ten participants (16%). It would be interesting to examine this ratio with a greater number of participants, and also with a sample of more experienced programmers.

Conclusions

It would appear that Weinberg's hypothesis was correct in that people of a certain personality type performed better at one aspect of the software development process. This being the case, further research is recommended in order to more precisely analyse what exactly these differences are. In addition, it may be advantageous for software companies to consider the strengths of their employees when assigning them tasks in the workplace. If some people, for whatever reason, are better able to perform code review tasks than others then it would seem prudent for software companies to capitalise on the strengths of their employees, and consider employees perhaps previously overlooked for this particular task.

Acknowledgements

This paper was written as part of the DIRC project (www.dirc.org.uk). DIRC is a UK-based interdisciplinary research collaboration concerned with the dependability of computer-based

systems, spread over five sites throughout the UK. The authors wish to thank colleagues and anonymous reviewers for useful comments and the sponsor (EPSRC) for funding this research.

References

1. Weinberg, G.M., *The psychology of computer programming*. 1971-1998, New York,: Van Nostrand Reinhold. xv, 288.
2. Pressman, R.S., *Chapter 19 - Software testing strategies, pp.654-658*, in *Software engineering : a practitioner's approach*. 1992, McGraw-Hill: New York. p. xxi, 793.
3. Shneiderman, B., *Software psychology: human factors in computer and information systems*. 1980: Cambridge, Mass. : Winthrop Publishers. 320.
4. Sommerville, I., *Software engineering*. 4th ed. International computer science series. 1992, Reading, Mass.: Addison-Wesley Pub. Co. xvi, 649.
5. Boehm, B.W., *Software engineering economics*. Prentice-Hall advances in computing science and technology series. 1981, Englewood Cliffs, N.J.: Prentice-Hall. xxvii, 767.
6. Furnham, A., *Personality at work : the role of individual differences in the workplace*. 1994, New York: Routledge. xxiii, 423.
7. Furnham, A., *Personality and individual differences in the workplace: Person-organization-outcome fit*. 2001.
8. Furnham, A., L. Forde, and K. Ferrari, *Personality and work motivation*. *Personality & Individual Differences*, 1999. **26**: p. 1035-1043.
9. Furnham, A., C.J. Jackson, and T. Miller, *Personality, learning style and work performance*. *Personality & Individual Differences*, 1999. **27**: p. 1113-1122.
10. Furnham, A. and T. Miller, *Personality, Absenteism and Productivity*. *Personality & Individual Differences*, 1997. **23**(4): p. 705-707.

11. Furnham, A., et al., *Do personality factors predict job satisfaction?* *Personality & Individual Differences*, 2002. **33**(8): p. 1325-1342.
12. Myers, I.B. and M.H. McCaulley, *Manual : a guide to the development and use of the Myers-Briggs Type Indicator*. 5th ed. 1985, Palo Alto, Calif.: Consulting Psychologists Press. xx, 420.
13. Capretz, L.F., *Personality types in software engineering*. *International Journal Human-Computer Studies*, 2003. **58**: p. 207-214.
14. Edwards, J.A., K. Lanning, and K. Hooker, *The MBTI and Social Information Processing: An Incremental Validity Study*. *Journal of Personality Assessment*, 2002. **78**(3): p. 432-450.
15. Smither, R.D., *The psychology of work and human performance*. 3rd ed. 1998, New York: Longman. xviii, 590.
16. Bishop-Clark, C. and D. Wheeler, *The Myers-Briggs Personality Type and its Relationship to Computer Programming*. *Journal of Research on Computing Education*, 1994. **26**(3): p. 358-70.
17. Devito, A.J., *Review of Myers-Briggs Type Indicator*, in *The Ninth Mental measurements yearbook*, J. Mitchell, Editor. 1985, Lincoln, Neb. p. 10301032.
18. Myers, I.B., *Introduction to type : a description of the theory and applications of the Myers-Briggs Type Indicator*. 12th ed. 1990, Palo Alto, CA: Consulting Psychologists Press. 31.
19. Weinberg, G.M., *The psychology of computer programming*. 2nd ed. 1998, New York,: Van Nostrand Reinhold. xv, 288.
20. Devito Da Cunha, A., "The Myers Briggs Personality Type as a Predictor of Success in the Code-Review Task." MPhil Dissertation, University of Newcastle, 2003.