Code Comprehension and MBTI Type

David Greathead
Centre for Software Reliability, University of Newcastle upon Tyne, Claremont Tower, Newcastle upon Tyne, NE1 7RU, United Kingdom
david.greathead@ncl.ac.uk

Abstract

This article outlines the ongoing work in relation to personality and programming ability. Previous work by the author focussed on code review and personality, discovering that certain personality factors were correlated with performance on a code review task. The current work is more concerned with code comprehension ability. In addition to the personality factors mentioned above, experience, cognitive style and the presence or absence of commented assertions are also to be assessed in relation to ability to comprehend code. The implications for building dependable systems will also be discussed.

1. Introduction

As has been noted before, large variations in programming performance have been observed [1]. In addition, it has also been noted that the various aspects of software development (such as design and testing) are very different in nature, and require different skills in order to complete them properly [2-4]. In order to build dependable systems, it would be foolhardy to proceed without first further examining some of these observations. A program could have the best design in the world and still be terribly flawed if it is poorly implemented or improperly debugged by a person prone to making errors brought about by (for example) their lack of attention to detail. Similarly, an individual could be the best able to debug a program in the available workforce, but unable to create a workable design. If it is possible that some general individual character types are in some way indicative of these abilities, then there is good reason to examine these types more closely in an experimental setting in order to properly capitalise on the strengths of the individuals within the workforce.

One such instance of these variations in performance was examined in previous research in an experimental setting [5]. Research was carried out with regard to analysing performance on a code-review task whilst also examining personality type, as measured by the Myers Briggs Type Indicator (MBTI) [6]. The current research is a follow on from this work. One of the factors noticed from the previous research was that many of the participants had trouble understanding what the Java code they were examining was supposed to be doing. That is, they had difficulty in understanding the code. Partly as a result of this, the aim of the current research is to examine the code comprehension ability of individuals.

2. Method

Participants will initially be presented with a piece of Java code, in this case a lift simulator program. They will then be presented with a number of questions concerning the function of the code. The questions are of such a nature that they cannot be answered simply by having an understanding of how lifts work. That is, certain foibles in the code will be exploited in such a way that participants must fully understand the code before they are able to answer the question. These questions will be multiple-choice in nature in order to ease assessment.

The experiment will be ‘between groups’ in its design in that the code given to participants will not be the same for all taking part. In this case, fifty percent of participants will be given a copy of the code which has had various assertions added into it in the form of comments. These comments will be missing for the remaining participants. In all other respects, the code will be identical for all participants. It is expected that the participants with the commented code will perform better at the task - what is of interest is *how much* better they perform.

All participants will also be given an MBTI questionnaire to complete. The results from this will then be compared with their performance on the task.

It is expected that certain types will perform better at the task than others, as per the findings from the previous work [5]. What will be interesting is whether or not some types of participant are better able to deal with the absence of the assertion comments than others.

The same will also be examined with regard to the *field dependence / independence* aspects of cognitive style [7]. Participants will be administered with an Embedded Figures Test in order to measure their level of field independence. This will then be compared to their performance on the task.

Finally, levels of experience will also be examined in that three different samples of participants will be used, each with their own level of experience. The first group of participants will be second year undergraduate computing science students (similar to those used for the code review research), the second group will be third year undergraduate computing science students, and the final group either professionals, or more experienced Masters level students. It is expected that the more experienced a person is, the better they will perform at the task. Again, of more interest is *how much* better they are. It is also intended to examine whether more experienced participants are better able to cope with the absence of the assertion comments than less experienced participants.

It is hypothesised that *intuitive*, *field independent*, experienced participants will be the best at coping with the absence of the assertion comments due to their experience and their ability to 'look at the bigger picture' while *sensing*, *field dependent*, inexperienced participants will become too focussed on small details and will perform more poorly than their peers in the absence of commented assertions. The experimental design is illustrated in Figure 1.

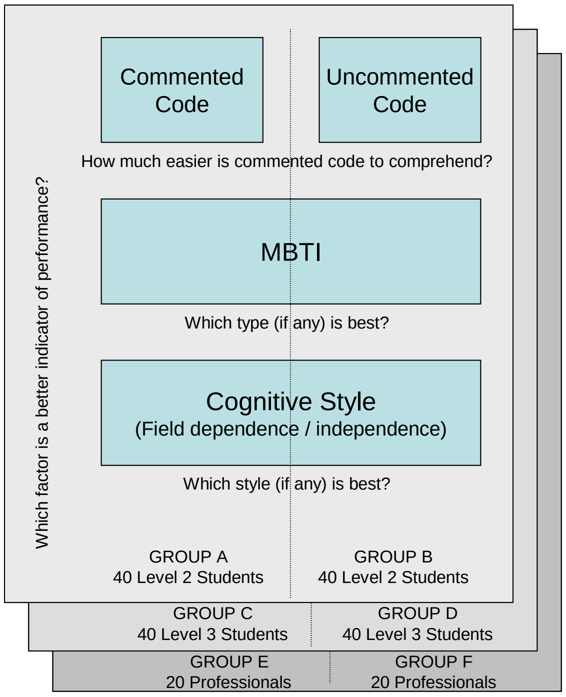

Figure 1. Diagram illustrating experimental design

3. Implications of the work

If carrying out this research leads to a greater understanding of the mental processes and personal characteristics important in the software creation process, this can only lead to the more appropriate use of employees for particular tasks, which in turn can only lead to better software products and therefore more dependable computer-based systems.

4. Acknowledgements

This paper was written as part of the DIRC project, a UK-based interdisciplinary research collaboration concerned with the dependability of computer-based systems (www.dirc.org.uk). The author wishes to thank colleagues and anonymous reviewers for useful comments and the sponsor (EPSRC) for funding this research.

5. References

[1] B. W. Boehm, *Software engineering economics*. Englewood Cliffs, N.J.: Prentice-Hall, 1981.

[2] R. S. Pressman, *Software engineering : a practitioner's approach*, 3rd ed. New York: McGraw-Hill, 1992.

[3] B. Shneiderman, *Software psychology : human factors in computer and information systems*. Cambridge, Mass.: Winthrop Publishers, 1980.

[4] G. M. Weinberg, *The psychology of computer programming*, Silver anniversary ed. New York: Dorset House Pub., 1998.

[5] A. Devito Da Cunha and D. Greathead, "Code Review and Personality: Is Performance Linked to MBTI Type?." University of Newcastle upon Tyne, 2004.

[6] I. B. Myers, M. H. McCaulley, and R. Most, *Manual, a guide to the development and use of the Myers-Briggs type indicator*. Palo Alto, Ca.: Consulting Psychologists Press, 1985.

[7] C. Bishop-Clark, "Cognitive style, personality, and computer programming," *Computers in Human Behavior*, vol. 11, pp. 241-260, 1995.