MBTI Personality Type and Student Code Comprehension Skill

David Greathead

Centre for Software Reliability
Newcastle University
David.Greathead@ncl.ac.uk

Keywords: Personality, MBTI, Code Comprehension

Abstract

This study aimed to examine the skill of participants in understanding the functioning of a piece of Java code by reading it. Their skill at this task was measured via a code comprehension task. The aim was to ascertain any relationships between this skill and their personality as measured by the Myers Briggs Type Indicator (MBTI).

The study was carried out with 74 participants, all of whom were from a computing science background and were of undergraduate, postgraduate or postdoctoral level, the majority being undergraduate students.

It was discovered that there was one significant interaction between personality as measured by the MBTI and code comprehension skill. Individuals who had a leaning towards Introversion on the Extroversion/Introversion preference were significantly better at the code comprehension task. None of the other three MBTI preferences yielded a significant result. This is discussed in the context of previous research.

1. Introduction

There is much evidence to support large variations in programmer productivity and ability within software development. In their classic paper Grant and Sackman (1967) observed programming performance differences of an order of magnitude with programmers of a seemingly similar background. Boehm reporting differences as anywhere between a factor of 10 and 30 (Boehm 1981). Shneiderman reported differences in programming performance of up to 100 to 1 in programmers of a similar background (Shneiderman 1980).

It was noted by Pressman that programmers performed differently on the same debugging task, even though those programmers had the same background (Pressman 1992). He speculated that there may be an ‘innate human-trait’ to explain this variation.

Weinberg too stated that each of the different stages in the programming process require different skills which suggests that different types of people would be better at some stages than others. He also suggested that personality may well be a factor in the individuals’ performance on these stages (Weinberg 1998).

Very broadly, the thinking behind the current research was to examine personality type as assessed by the Myers Briggs Type Indicator (MBTI), and examine the influence of this on performance on a code comprehension task. Reasons for these choices are discussed below. This was done with a view to begin to understand some of the factors at play which may account for the large differences in

programmer productivity reported in the above studies. Based on the findings from this work, the aim would then be to better understand some of the differences between individuals and their skill with regard to some of the tasks involved in the software development process and why these differences may exist. Once done, the knowledge gained could then be used to feed back to teaching methods to both inform potential students about some of the difficulties different types face when learning to program, as well as to perhaps even tailor teaching methods to counter some of the difficulties certain types would be likely to face. This knowledge may also be helpful in constructing more effective programming teams.

The MBTI was chosen to be used in this research partly as it is used in the computing industry. As a result of this it appears numerous times in research literature, see for example Bishop-Clark and Wheeler (1994) and Bishop-Clark (1995) for a partial review of the literature. Additionally, this code comprehension research is a direct follow on from some previous research which examined code review skill and MBTI type (Devito Da Cunha 2003; Devito Da Cunha and Greathead 2007).

1.1. The MBTI

The MBTI was originally developed by Katherine Cook Briggs and Isabel Briggs Myers in the 1940s. It was based on the personality theory of Carl Gustav Jung (Jung 1976). Jung initially differentiated people into the two types of Introversion and Extroversion which concerned a person's general *attitude*. This distinction pervades many different personality questionnaires that are still in use today. It was Jung's belief that people have both introverted and extroverted aspects to their personalities and that they have the ability to act in a way appropriate to each type. He also said that people would have a *preference* towards one type or the other.

Jung then went on to add in two more pairs of preferences, these being the Functional Types of Thinking and Feeling as well as Sensation and Intuition. He stated that the Thinking or Feeling distinction was what enabled individuals to make their decisions on a day to day basis. He called these 'Rational'. The Sensation and Intuition functions were the ways in which people gathered their information from the outside world. He called these 'Irrational'.

Following on from Jung's theories, Katherine Cook Briggs and Isabel Briggs Myers began to develop their method of measuring these different aspects of a person's personality. They also took some of the work by Jung, which while not directly part of his personality theory was concerning the need people have to make judgements and perceive information from the outside world. They made this into the fourth pair of preferences on the MBTI which would mainly be used in order to differentiate between whether the person has a preference for the TF attitude (Judging) or the SN attitude (Perceiving).

Each person taking the MBTI will find themselves as one or the other of each of these four types. This being the case, each person will be presented with a four letter type, which summarises their personality type as decided by the MBTI. For example, a person who was Extrovert, Intuitive, Feeling and Perceiving would be reported as ENFP (Intuition is represented as iNtuition as the 'I' is already used to represent Introversion). The opposite type to this would be ISTJ. Naturally, these letter types combine to make the 16 types familiar to those who use the MBTI (Myers and McCaulley 1998).

1.2. Blurring the MBTI Types

In scoring the MBTI, various numbers are returned along with the actual letter type. While these numbers are not, strictly speaking, the strengths of the preferences, they may well have some value. For instance, the numbers are used in the follow up sessions which sometimes accompany the MBTI.

For simplicity, from this point onwards the numbers returned for each of the four preferences will be referred to as a 'strength of preference' or as a point on a scale although the author accepts that this is not how the MBTI was designed. As is pointed out in the MBTI manual 'The indices EI, SN, TF and JP are designed to point in one direction or the other. They are not designed as scales for measurement of traits or behaviours' (Myers and McCaulley 1998). However, it seems reasonable to conclude, based on purely practical observation, for example, people who answer each of the EI questions in the I direction may be, to the outside observer, more Introverted than someone who answers 51% of those questions in the I direction.

Therefore, despite the fact that the MBTI was not designed to be used to produce scales, it is the author's belief that, after transforming the numbers returned from the questionnaire into four bipolar scales, it is possible to extract some useful information from these numbers and for the purposes of analysis each of the four preferences will also be treated as scales where it is appropriate to do so. It should also be pointed out that studies in the past have carried out comparisons between MBTI and other personality questionnaires based on Trait Theory with results showing similarities between the various scales. In doing so, they also treat the numerical values returned by the MBTI as scales. Some studies do not specify that they convert the numerical values to make them into equal scales but it is assumed that all do given that without this, the raw scales are not all scored out of the same amount, even within a scale. Saggino, et al (2001) state that; 'it is possible to transform these indexes into continuous scores which are a linear transformation of the preference scores'. Furnham also treats the MBTI preferences as continuous scales in order to carry out comparisons between the MBTI and other personality scales, (Furnham 1996; Furnham et al. 2001). For example, with the raw scores, one side of a scale may be out of 76 points while the other end of the scale is out of 59.

The decision to use the numerical data from the MBTI was also partly based on the author's own experiences as he noted that his type often changed in three of the four letter types between measurements and observation of the numbers returned revealed that he was in fact balanced on three of these four scales. This results in the author being comfortable in 12 out of the possible 16 types. Clearly however, he may not display characteristics as stereotypically representative of one of these types when compared to another person who consistently reports as the same type each time the test is taken.

1.3 Background to the Study

Work by Capretz (2003) states that the area of software engineering is dominated by Introvert types. He argues that certain aspects of software engineering, which have a highly technical or mathematical content may be better suited to Introverts. It should be noted that he also states that certain types such as Introverts or Thinkers may be regarded as the typical programmer types, but that these associations may not be as true to this day, given the prevalence of software in everyday life, and the changing nature of software development. He stresses that ability to communicate and work well in teams is more important in the modern software industry. He cites communication with the user and discussion of problem solving approaches within a programming team as areas where Introversion will be less of a help. In addition, Chandler et al (2003) state that the undergraduate student population of computing students is dominated by Introverted, Sensing and Judging types. They studied three UK universities and found that these types were greatly over-represented when compared to the general population.

Despite Capretz's view, some tasks still exist in modern software engineering where communication skills are less essential. Code review, debugging or code comprehension, for example may all be regarded as solitary activities under certain circumstances. Given the large percentage of Introvert types in the software industry, it seems that there is some form of self-selection taking place with regard to career choice. It also seems reasonable to assume that this choice is not a blind decision on the part of those concerned. That is, there may well be something in the nature of the work which draws introvert types.

Bishop-Clark and Wheeler (1994) suggest that, given the nature of 'computer programming', Introvert types would be expected to perform better than Extroverts with regard to programming related activities. Kagan and Douthat (1985) found that Introverted participants in their study performed significantly better on an introductory FORTRAN course. In a study relating to programmer personality, Turley and Bieman (1995) discovered that 90% of their industry sample were Introvert types.

Bishop-Clark and Wheeler (1994) also state that Intuitives are good at viewing a problem in different ways which encompass the 'big picture'. Capretz (2002) states that 'Intuitive types have a greater interest in dealing with material which is abstract and symbolic' (p135). In Capretz' study (2003) he discovered that Intuitive types were overrepresented in his sample of professional programmers when compared with the general population. Intuitive Thinkers in particular were the most overrepresented two-letter type in the sample. He goes on to state that, according to MBTI theory, NTs are more likely to be creative than STs because they see beyond simple facts, but are in tune with relationships, and identifying underlying principles. In this way, it could be argued that Intuitives, Thinkers, and NTs in particular are more likely to be able to perform well on certain programming-related tasks.

Bishop-Clark and Wheeler (1994) state that the nature of the Thinking type as defined by the MBTI is very well matched to some tasks involved in software development, specifically with regard to problem-solving related areas and anything which requires standardised procedures, working with defining constraints and tasks requiring impersonal, sequential and logical analysis. This, they point out, is the very definition of the Thinking type as defined by the MBTI. In the Turley and Bieman (1995) study of programming professionals, 85% of their sample were of the Thinking type and in Capretz' work (2003) 81% of their sample of programming professionals were Thinking. Similarly, Chandler et al. (2003) found 86% of their sample of computing students were of the Thinking type. Clearly a large amount of self-selection is taking place in the programming areas, and the theory suggests that Thinking types would be better suited to tasks involved which are related to logical thinking (Myers, 1998).

Previous work carried out was aimed at establishing a basic link between MBTI type and an easily assessable aspect of the programming process (Devito Da Cunha 2003; Devito Da Cunha and Greathead 2007). This previous work examined undergraduate student skill on a code review task and related performance on the task with MBTI type. The task required participants to read through four printed pages of Java code and highlight any bugs they found. Their performance on this task was then related to the MBTI results. It was discovered that the only significant result was when comparing Intuitive Thinking (or NT) types with other types. NT types were significantly better at the code review task, illustrating that some relationship does exist between MBTI type and some aspect of the programming process.

The current research was a direct follow on from the code review work (Devito Da Cunha 2003; Devito Da Cunha and Greathead 2007). The aim was to examine the theories and findings in the above literature. As a result, and based on the existing literature, the following four hypotheses were developed.

1.4 Hypotheses

Based on previous research and the literature mentioned above, four hypotheses were developed. These were:

- Introverts will perform significantly better than Extroverts on the code comprehension task.
- Intuitives will perform significantly better than Sensors on the code comprehension task.
- Thinkers will perform significantly better than Feelers on the code comprehension task.
- Intuitive Thinkers will perform significantly better than other types on the code comprehension task.

2. Methodology

There were 74 participants. All were studying on the Computing Science or Software Engineering programmes or had relevant programming experience. Participants were unselected for age and sex, however due to the self selection found amongst computing science students, the majority of participants were aged between 19 and 23 (range 19-36). Most were male (65, 88%) with nine females (12%). Participants were recruited in a number of ways, these being:

- Approached at the end of their normal lectures by the experimenter who talked to them as a group.
- E-mailed with details of the experiment by an appropriate member of staff.
- By word of mouth from previous participants in the experiment.
- Via ‘networking’, i.e. friends and associates of the experimenter would ask people they knew if they wanted to take part.
- Told about the experiment by lecturers during their normal lectures.

The code comprehension task was developed over a number of sessions by a team of experienced programmers. Initially, a pre-existing program was selected from the open source community based on its complexity and length, which needed to be appropriate for the research. The program was written in Java (the language taught to all computing students at the university) and was a lift simulator program. The questions for the task were developed by the programmers by first familiarising themselves with the code, then having a group discussion with the others to decide upon the form which the questions should take, and their rough level of difficulty. Some rough questions were developed in the group setting before the programmers developed their own questions over a period of a few days. After this time, a further group session took place, where all of the questions were discussed, along with the correct answers, and plausible incorrect answers for the multiple choice questions.

The next step was to pilot the questionnaire to gauge the level of difficulty. A set of questions was selected and tested on a group of volunteers. The results from this were then discussed with the programmers and the final questions were chosen, with minor changes being made to some of these.

The test was designed in order to measure a meaningful skill while not being an impossibly long or difficult task. The task began with an introductory page which gave instructions for completing the task as well as emphasising that the answers to the multiple choice questions should not be based on assumptions about how lifts work in real life. The first two sections of the task consisted of a number of multiple choice questions which were relatively easy and aimed at helping the participants begin to understand the code. The final part of the comprehension task was more difficult and aimed at discovering the individuals who really understood the program. The maximum score possible for the task was 27. This questionnaire was accompanied by four screenshots of the program in operation which were directly related to four of the questions in the final part of the task.

Due to the fact that the experiment took approximately two hours, participants were paid £20 for taking part in the research. Three different groups of participants were approached, these being second year undergraduate students, third year undergraduate students and others with relevant programming experience. No participants were excluded from the study.

Participants were required to complete the MBTI (Myers, 1998), specifically the Step 1 version of the European English Edition, which was accompanied by the self-scorable version of the answer book. Then, a short video demonstrating the lift simulator program running was shown in order to aid participants' understanding. This was to be shown via data projector. The excerpt of video was selected in such a way that participants could not determine the answers to the questions in the task based on what they saw in the video, rather it could only be used as an aid to develop a broader understanding of the program. The purpose of the clip was to aid participants' understanding of the program and to demonstrate what the graphical user interface (GUI) looked like. The video clip lasted for one and a half minutes. Again at this point, participants were reminded that the lift simulator program did not necessarily represent how lifts operate in the real world and was a flawed simulation which often exhibited counter-intuitive behaviour. They were reminded that in order to answer the questions correctly, they would need to refer directly to the code itself.

The experimenter encouraged the participants to make a serious attempt at all questions and not to give up too easily. They were informed that they would need to spend approximately an hour if not more on the code-comprehension task. The rooms were always booked for three hours and participants were told of this fact and that if they wanted more time to complete the task they could have it. They were also informed that some of the questions were quite difficult and that they should not simply give up if they could not immediately see the answer. This was done in order to attempt to motivate participants to apply themselves to the task rather than simply giving up when they encountered a difficult question.

The data was gathered over as small a number of sessions as possible and individual sessions usually contained people of a similar background. This aided in minimising any information regarding the task passing from participants who had completed the task to those yet to do it. Participants are more likely to have contact with their classmates than with people from another year of study. Participants were also required to return all task-related materials in order to minimise the potential of these materials coming into the possession of participants yet to attempt the task.

The task was also developed in such a way as to be possible to carry it out without the use of a computer. While this may be a very specific task, activities such as code reading (and therefore the skill to understand code by reading it) does still have value today (Siy and Votta, 2001). A further advantage of having a pen and paper test is that the conditions under which the test is administered

can be more easily controlled. It is much easier to restrict useful internet access when the participants are in a room without computer access (although even this is harder to control with modern mobile telephones). Yet another advantage of a pen and paper test is purely a practical one in that is makes the data collection process itself easier as, for example, large rooms in which to run the experiment are much easier to obtain if large amounts of computers are not required.

The code comprehension task was comprised of a number of separate sections based around a lift simulator program, written in Java. The code itself consisted of 30 A4 pages of Java. The task began with an introductory page which gave instructions for completing the task as well as emphasising that the answers to the multiple choice questions should not be based on assumptions about how lifts work in real life. This questionnaire was accompanied by four screenshots of the program in operation which were directly related to four of the questions. The questionnaire and the code used can be found at www.minastirith.co.uk/research.

The dependent variables for the study were as follows:

- Correct answers from the multiple choice questionnaire regarding code comprehension.
- MBTI type and preference strength.

3. Results

Firstly, differences between the second year students, third year students and postgraduate students were examined. There was found to be no significant difference between these groups with regard to score on the task and it was decided that it would be possible to treat all the participants as one homogeneous group for the purpose of statistical analysis.

In order to examine the interaction between MBTI and the results from the code comprehension task, the first step was to compare each letter type of the MBTI with performance on the task. This was done using a series of t-tests. A two-letter analysis was also carried out. The results from these tests are shown in Table 1, below.

MBTI	N	Mean	SD	MBTI	N	Mean	SD	t	p
E	40	16.93	3.452	I	30	18.80	2.592	2.493	0.008
S	30	17.20	3.044	N	40	18.13	3.345	-1.189	0.119
T	51	17.75	3.149	F	19	17.68	3.528	0.070	0.473
J	28	17.56	3.271	P	42	17.90	3.230	-0.556	0.580
Non-NT	42	17.05	3.177	NT	30	18.53	3.093	1.978	0.026

Table 1. T-tests and means comparing MBTI type on the code comprehension task (DF=68).

However, given that one might regard the letter-type distinction as somewhat blunt, it is also worth examining a correlation based on the strength of preference on the MBTI as returned by the MBTI questionnaire. In order to assess this, the maximum scores possible for each of the scales on the MBTI were calculated. Once this had been done, it was then possible to carry out a valid correlation analysis. Results for these tests are summarised below.

MBTI	Pearson's r	p
EI	-0.347	0.003
SN	-0.127	0.148
TF	0.009	0.471
JP	0.002	0.988

Table 2. Table of correlation results for MBTI and code comprehension (N=70).

3.1. Extroversion/Introversion

The first letter type to be examined was Extroversion/Introversion (E/I). For this scale, it was expected that Introverts would perform better on this task than Extroverts and as such it is appropriate to use the one-tailed level of significance. Due to the number of tests used, it is necessary to carry out a Bonferroni adjustment on these t-tests. This leads to a significance threshold of 0.01. As can be seen, there were significant differences between Extroverts and Introverts on the code comprehension task. As expected, Introverts performed better at this task than Extroverts.

For the correlation, on the E/I scale 100 would represent maximum Extroversion while 0 would represent maximum Introversion with 50 being neither Extrovert nor Introvert. As can be seen from the results, the correlation for the E/I scale was negative which indicates that as someone moves to the upper end of the scale (becoming more Extrovert) then they perform worse on the code comprehension task. Again, a Bonferroni adjustment on the correlations leads to a significance threshold of 0.013. Clearly the EI scale is significantly correlated with the score on the code comprehension task. This would suggest that using the E/I scale is a good indicator of likely performance on this task, and thus an indicator of code comprehension skill (assuming all other variables to be equal).

3.2. Sensor/Intuition

The next step in analysing the relationship between MBTI type and code comprehension skill was to carry out the same tests on each of the other three scales. The next preference to examine was Sensor/Intuition (S/N). The results of the t-tests were not significant. Nonetheless, to capitalise on the extra information contained within the numerical values returned from the MBTI, correlations were also carried out. It was clear from looking at the results from these correlations, that while the directions of the correlation was in the expected direction (a high score on the S/N scale indicating a Sensing preference), the correlations were weak, and not significant. It must be concluded therefore that the Sensing/Intuition scale has little impact on code comprehension skill, as measured by this test.

For this study, it was expected that Intuitive types would perform better on the task given that Intuitive types are generally regarded as being able to view problems in different ways and to see the 'big picture' which was specifically required given the complexity of this task (Bishop-Clark and Wheeler, 1994; Myers, 1998). This was however not the case. Conversely, Bishop-Clark and Wheeler (1994) discovered that Sensing types were more likely to score well on an introductory programming class, but this relationship was also not replicated here. It seems to be, therefore, that the nature of this task is different to those tasks involved in a basic programming task. It also seems to be that the code comprehension task here was of such a nature that neither the ability to see the 'big picture' and deal with problems in a variety of ways was an advantage to this task. In this way, one might argue that this task is neutral with regards to Sensing/Intuition preference.

Additionally, this task is quite specific in its nature, while a general introductory programming course, as used by Bishop-Clark and Wheeler (1994) will no doubt cover a variety of concepts and skills, but perhaps at a relatively basic level. This would account for the Sensing preference being an advantage in that context.

3.3. Thinking/Feeling

The third preference to examine was that of Thinking/Feeling. It was expected that Thinking types would be better at the comprehension task based upon previous research (Bishop-Clark and Wheeler, 1994; Devito Da Cunha and Greathead, 2004; Devito Da Cunha and Greathead, 2007). T-tests were carried out as in the case of the previous two preferences to examine this scale with regard to performance on the code comprehension task. The results from this test was however non-significant. The T/F letter type on the MBTI has no relationship to participants' performance on the code comprehension task. This is a somewhat unexpected result in that it was believed that thinking types would be better at this task. It is worth noting, however that there is clearly some self-selection taking place before this point given that 51 participants had the Thinking letter type and only 19 had the Feeling type. This being the case, the value of using the numbers obtained from the MBTI to carry out correlations is even more apparent. However, the results from these correlations indicate that the T/F scale has no relationship to participants' performance on the code comprehension task with all of the correlations scoring virtually zero. This is contrary to expectations and is discussed in more detail below.

3.4. Judging/Perception

For completeness, the same tests were carried out for the J/P preferences but again, no significant results were found for this preference.

3.5. Two Letter Type Analysis

In addition to the above, it was also decided to compare performance on the code comprehension task based on two letter types. It was reasoned, based on previous work (Devito Da Cunha and Greathead 2007) that the combination of the S/N and T/F scales would be the important combination. An Analysis of Variance was carried out comparing ST, SF, NT and NF types but did not yield a significant result.

As with previous work, it was also decided to compare NT types to non-NT types. It was anticipated that NTs would be better than other types given their ability to infer information and to think logically (Capretz, 2003, Myers 1998), thus a one-tailed significance threshold is used. The results of the NT versus non-NT t-test were shown in Table 1.

As can be seen, the NT types were not significantly better at this task than non-NT types. This contrasts with previous work regarding code-review skill and indicates that the two tasks are indeed different and require different skills. It is interesting to note that the Computing Science population tends to have a disproportionate number of NT types when compared to the general population (in this sample NTs accounted for approximately 43% of types which compares with the figure of approximately 15% of a college sample of over 32,000 people reported in the MBTI manual (Myers and McCaulley 1985).

4. Discussion

When considering the influence of MBTI on performance on the code comprehension task, only one of the factors showed any significant impact, and that was the Extroversion/Introversion preference, with Introverts performing significantly better on this task. It was also indicated that the more Introvert an individual was the better they performed on the code comprehension task.

This result is quite clear and suggests that the MBTI is a good indicator of how well they performed on this task. This is clearly therefore a very interesting and significant result. If all other factors, such as level of education and training were constant one would still expect Introverts to perform better on this task than Extroverts. It could be that Introverts simply are more comfortable working on this kind of task as in this case, it was a solitary activity and not only did it not require interaction with other participants, but actually demanded no interaction with others in order to prevent participants helping each other. Is it possible that the more Extrovert individuals would perform better on such a task if they were allowed to communicate with other people, which indeed may be how this kind of situation would be dealt with in reality. However, in the scope of this study, which involved individual participants working alone on the task, Introvert participants did perform significantly better than Extrovert ones did.

This result confirms the theories of Bishop-Clark and Wheeler (1994) and the findings of Kagan and Douthat (1985) that Introversion/Extroversion is a significant factor with regard to certain programming related tasks. It also perhaps explains to some extent the overrepresentation of Introverts found in the programming area. It could simply be that most programmers are Introverted because Introverts are drawn to the area due to the fact that they enjoy and are skilled at some of the tasks involved.

As Capretz (2003) states, the notion of programming being a solitary activity is less relevant today than in the past. With software having spread to so many aspects of everyday life, computer programs are not always concerned with highly mathematical content. In addition as Capretz states, most modern programming problems are tackled by teams of programmers and in those cases what it required is a team with a mix of psychological types. Nonetheless, some activities which are quite solitary in nature and are still required. Programming as an activity consists of many phases, each requiring a variety of skills (Weinberg, 1998). It is therefore worth examining relationships between, among other things, personality especially given this result, and that of previous work (Devito Da Cunha 2003; Devito Da Cunha and Greathead 2007).

With regard to the other aspects of the MBTI, no other single variable had any significant relationship with performance on the code comprehension task. Previous research suggested that the S/N scale would be most likely to have an influence on programming skill (Bishop-Clark and Wheeler 1994), but the findings here do not support this. In their study, they found that students with a Sensing preference performed significantly better on an introductory programming course than those with an Intuitive preference.

It was also expected that the Thinking/Feeling scale would bear a significant relationship, with Thinking types performing better on the task, but again this was not the case. It was anticipated that thinking types would be aided in their preference for logical decision making when building up a working mental model of the program by reading the code. It was anticipated that there would be some overlap between their preference for basing their decisions on thinking and their skill in tracing the required information through the program in order to answer the questions in the task correctly. It

seems however that this is not the case and the Thinking/Feeling preference alone bears no relationship to code comprehension skill.

It is possible that this type of task (unlike some other tasks involved in the programming process) does not require too much logical decision making. Perhaps, as suggested by the Extroversion/Introversion results, it is more that participants are simply required to understand to a basic level and the ability to perform comes from being more applied to the task, or more likely to focus on the job at hand, rather than becoming distracted by the immediate social situation (such as talking to nearby people rather than working on the task). This would explain why Extroversion/Introversion had a significant effect while Thinking/Feeling did not.

The two letter comparison of NT compared with non-NT was not significant. This is in contrast with previous work regarding skill on a code-review task (Devito Da Cunha and Greathead, 2004; Devito Da Cunha and Greathead, 2007). In the previous work, NT types were significantly better than non-NT types when reading code and locating the errors within that code. While the code comprehension task of the current study and the code-review task of the previous study may seem similar in nature, the discrepancy in the results suggests that the skills involved are in fact different.

5. Conclusion

The results from the Extroversion/Introversion scale analyses make it abundantly clear that MBTI type does indeed have a relationship with performance on code comprehension skill as it was measured by this task. This, combined with the fact that other significant relationships between MBTI preferences and other aspects of programming related skill have previously been discovered (for example Kagan and Douthat, 1985; Bishop-Clark and Wheeler, 1994; Devito Da Cunha and Greathead, 2004; Devito Da Cunha and Greathead, 2007) illustrates that there is value in examining personality with regard to skill in various software development tasks.

As Capretz (2003) states; ‘Due to the diverse nature of software engineering, it is widely believed that no personality instrument will ever accurately predict success in this field,’ p.214. He also states that it takes a variety of skills and personalities to solve the various problems related to software engineering and that companies should consciously attempt to construct teams of software engineers with diverse and complimentary personalities. This is not mutually exclusive with the findings from the current and previous personality research in this area. Given that it has been shown that certain skills related to the programming process are associated with certain personality preferences, if a rich enough set of observations can be made it may be possible to construct teams of programmers specifically designed to deal with particular short term problems. This could be based on not only personalities which are complimentary to each other to make harmonious and diverse teams, but also ones which have a spread of types specifically chosen to deal with the individual problems the teams will face for that particular task.

6. Acknowledgements

This paper was produced in association with the EPSRC Platform Grant on Trustworthy Ambient Systems (TrAmS) and was based on research originally carried out as part of the DIRC project (http://www.dirc.org.uk/). Thanks to Cliff Jones for making the research possible and to Marian Petre and Peter Andras for giving the push to produce this paper. Thanks also to the anonymous reviewers for their helpful feedback.

7. References

Bishop-Clark, C. (1995). "Cognitive style, personality, and computer programming." Computers in Human Behavior **11**(2): 241-260.

Bishop-Clark, C. and D. D. Wheeler (1994). "The Myers-Briggs Personality Type and Its Relationship to Computer Programming." Journal of Research on Computing in Education **26**(3): 358.

Boehm, B. W. (1981). Software engineering economics. Englewood Cliffs, N.J., Prentice-Hall.

Bishop-Clark, C. (1995). "Cognitive style, personality, and computer programming." Computers in Human Behavior **11**(2): 241-260.

Bishop-Clark, C. and D. D. Wheeler (1994). "The Myers-Briggs Personality Type and Its Relationship to Computer Programming." Journal of Research on Computing in Education **26**(3): 358.

Capretz, L. F. (2002). "Implications of MBTI in software engineering education." SIGCSE Bull. **34**(4): 134-137.

Capretz, L. F. (2003). "Personality types in software engineering." International Journal of Human-Computer Studies **58**(2): 207-214.

Chandler, J., J. Carter and I. Benest (2003). Extravert or Introvert? The Real Personalities of Computing Students. 4th Annual LTSN-ICS Conference. NUI Galway.

Devito Da Cunha, A. (2003). The Myers Briggs Personality Type as a Predictor of Success in the Code-Review Task. Psychology. Newcastle Upon Tyne, University of Newcastle Upon Tyne.

Devito Da Cunha, A. and D. Greathead (2004). Code Review and Personality: is Performance Linked to MBTI Type? (Technical Report). Newcastle, School of Computing Science, University of Newcastle Upon Tyne.

Devito Da Cunha, A. and D. Greathead (2007). "Does personality matter? An analysis of code-review ability." Communications of the ACM **50**(5): 109-112.

Furnham, A. (1996). "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.

Furnham, A., C. J. Jackson, L. Forde and T. Cotter (2001). "Correlates of the Eysenck Personality Profiler." Personality and Individual Differences **30**(4): 587-594.

Grant, E. E. and H. Sackman (1967). "An Exploratory Investigation of Programmer Performance Under On-Line and Off-Line Conditions." Human Factors in Electronics, IEEE Transactions on **HFE-8**(1): 33-48.

Jung, C. G. (1976). Psychological types. Princeton, N.J., Princeton University Press.

Kagan, D. M. and J. M. Douthat (1985). "Personality and learning FORTRAN." International Journal of Man-Machine Studies **22**(4): 395-402.

Myers, I. B. (1998). MBTI manual : a guide to the development and use of the Myers-Briggs Type Indicator. Palo Alto, Calif., Consulting Psychologists Press.

Pressman, R. S. (1992). Software engineering : a practitioner's approach. New York, McGraw-Hill.

Saggino, A., C. Cooper and P. Kline (2001). "A confirmatory factor analysis of the Myers-Briggs Type Indicator." Personality and Individual Differences **30**(1): 3-9.

Shneiderman, B. (1980). Software psychology : human factors in computer and information systems. Cambridge, Mass., Winthrop Publishers.

Turley, R. T. and J. M. Bieman (1995). "Competencies of exceptional and nonexceptional software engineers." Journal of Systems and Software **28**(1): 19-38.

Siy, H. and L. Votta (2001). Does The Modern Code Inspection Have Value? IEEE International Conference on Software Maintenance, Florence, Italy.

Weinberg, G. M. (1998). The psychology of computer programming. New York, Dorset House Pub.