The Influence of Human Factors on Programming Performance: Personality, Programming Styles and Programming Attitudes

by

Xuechao Li

A dissertation proposal submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 6, 2017

Keywords: Performance, Programming Attitudes, Programming Styles, Personality, Empirical Studies

Approved by

Dr. Cheryl D. Seals, Chair, Associate Professor of Computer Science and Software Engineering
Dr. Alvin Lim, Professor of Computer Science and Software Engineering
Dr. Dean Hendrix, Associate Professor of Computer Science and Software Engineering
Dr. Robert Thomas, Professor of Industry Engineering
Dr. Jeff Overbey, Assistant Professor of Computer Science and Software Engineering

Abstract

The programming performance has been studied over several years. Researchers and scientists utilized various optimization technologies on algorithms and computer architectures to improve the performance. But, to date, few studies focus on the impact of human factors on the programming performance. In this study, we investigate the influence of human factors on the programming performance based on Mayer-Briggs Type Indicator (MBTI) personality, programming attitude and programming styles. Although some researchers have investigated the effects of personality based on the Five-Factor model on programming styles, two problems are not resolved: (1) Five-factor personality model does not theorize what goes inside people's heads and focuses on actual people's behaviors instead of the cognitive theory; (2) the programming styles were not validated and are out of date. To improve this research work, a theoretical personality model-- Myers–Briggs Type Indicator – is adopted. In addition, the programming styles have been updated since 2006 and validated using statistical metrics such as Cronbach's Alpha. Finally, a new programming factor-- programming styles-- are added into our investigation. The objective of this proposal is: (1) to identify which human factors play a positive/negative role in programming performance; (2) to study the relationship among personality, programming styles and programming attitudes. The author firstly distributes three questionnaires on personality, programming attitudes and programming styles to students in department of computer science and software engineering at Auburn University. Three surveys towards programming will be measured via the self-assessed method. The programming

performance consists of: (1) run time from participants' code; (2) grades of projects. The analysis, such as Pearson Correlation analysis and linear regression analysis, will be applied to investigate the links among personality, programming styles and programming attitudes.

Acknowledgments

I would like to thank Dr. Cheryl Seals for her dedication during the proposal development phases and for serving as my Chair. Thanks to Dr. Jeff Overbey, Dr. Alvin Lim, Dr. Dean Hendrix, and Dr. Robert Thomas for serving on my committee.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

IPIP	International Personality Items Pool
EPI	Eysenck Personality Inventory
LSQ	Learning Style Questionnaire
MBTI	Myers Briggs Type Indicator
FFM	Five-Factor Model
MTMM	Multi-Trait, Multi-Method
PCA	Principal Component Analysis
MSA	Measure of Sampling Adequacy

Chapter 1 Introduction

1.1 Motivation

The software development progresses through various phases, which ultimately leads to the final product. Among the phases, the coding's performance and efficiency has a significant impact on the progress of the final software. If the software is a large and complex system such as Ubuntu, it may be divided into subsections which will be integrated later. Sometimes researchers use coding productivity to evaluate the efficiency/performance of programmers, and they notice that the programming style is one of the dominant factors in coding work. Pressman [1] conducted the empirical experiment and made the conclusion that programmers' coding performance was on a different level for that same task although they have the same academic background. After analyzing all variances of the experiment, the author found that different human traits may result in the coding performance difference.

Also, the performance is one of the most important metrics in evaluating the code. In this study, we use both the running time of the code and the corresponding grades based on the rubrics, to define the performance with the following considerations: (1) the running time can objectively measure the quality of code written by programmers; and (2) from the perspective of human factors, we still need to evaluate whether the code, written by one programmer, can be easily read by other programmers if this coding work needs to be transferred between members in a big team. From the author's experience, although there is a similar performance in the running time among code samples, some of those samples are

difficult for other programmers to understand or optimize. So, the rubric can perfectly detect and evaluate the performance from the perspective of human factors.

Some theorists noted that there were surprisingly large variations in individual productivity and accuracy while executing parts of the software development process [15]. We know the choice of programmers makes a significant performance impact, but we have little insight into how that choice impacts programming performance [39].

To date, human traits or individual characteristics can be evaluated through either Big Five-Factor model(FFM) or Myers-Briggs Type Indicator model(MBTI). For FFM model, some personality tests such as IPIP (International Personality Items Pool) [2] is widely used in academia. The Big Five-Factor traits are Openness, Conscientiousness, Extroversion, Agreeableness, and Neuroticism (OCEAN), and each trait is briefly introduced in the following sections [43]:

Openness - People who like to learn new things and enjoy new experiences usually score high in openness. Openness includes traits like being insightful and imaginative and having a wide variety of interests.

Conscientiousness - People that have a high degree of conscientiousness are reliable and prompt. Traits include being organized, methodic, and thorough.

Extroversion - Extroverts get their energy from interacting with others, while introverts get their energy from within themselves. Extraversion includes the traits of energetic, talkative, and assertive.

Agreeableness - These individuals are friendly, cooperative, and compassionate. People with low agreeableness may be more distant. Traits include being kind, affectionate, and sympathetic.

Neuroticism - Neuroticism is also sometimes called Emotional Stability. This dimension relates to one's emotional stability and degree of negative emotions. People that score high on neuroticism often experience emotional instability and negative emotions. Traits include being moody and tense.

There has been numerous research studies carried out on human traits influencing work performance. Adrian *et al.* [45] investigated the relationship between personality, learning style and work performance based on responses of over 200 telephone sales staff. The Eysenck Personality Inventory (EPI) and Honey and Mumford's Learning Style Questionnaire (LSQ) were used to evaluate human traits. Results showed that personality variables (extraversion, neuroticism) and certain learning styles (reflector, pragmatist) were statistically significant predictors of the rated performance.

In addition to the Five-Factor personality model, Myers-Briggs Type Indicator (MBTI) is another popular model which is widely used in academia. The Myers-Briggs Type Indicator measures preferences on four scales derived from Jung's Theory of Psychological Types. People are classified in terms of their preference as the following [43]:

- **Introversion (I)** (interest flowing mainly to the inner world of concepts and ideas) or **Extroversion (E)** (interest flowing mainly to the outer world of actions, objects, and persons);

- **Sensing (S)** (tending to perceive immediate, real, practical facts of experience and life) or

  Intuition (N) (tending to perceive possibilities, relationships, and meanings of experiences);
- **Thinking (T)** (tending to make judgments or decisions objectively and impersonally) or

  Feeling (F) (tending to make judgments subjectively and personally);
- **Judging (J)** (tending to live in a planned and decisive way) or

  Perceiving (P) (tending to live in a spontaneous and flexible way).

All possible permutations of preferences in the four dichotomies above yield sixteen different combinations, or personality types, representing which of the two poles in each of the four dichotomies dominates in a person; thus, defining sixteen different personality types. In Table 1, each personality type can be assigned a four-letter acronym as corresponding combinations of preferences. Since one specific type will be selected from each category, the final personality is a combination of four personality types. Figure 1 describes this selection process.

Table 1. The Sixteen Personality Types [43]

ESTJ	ISTJ	ENTJ	INTJ
ESTP	ISTP	ENTP	INTP
ESFJ	ISFJ	ENFJ	INFJ
ESFP	ISFP	ENFP	INFP

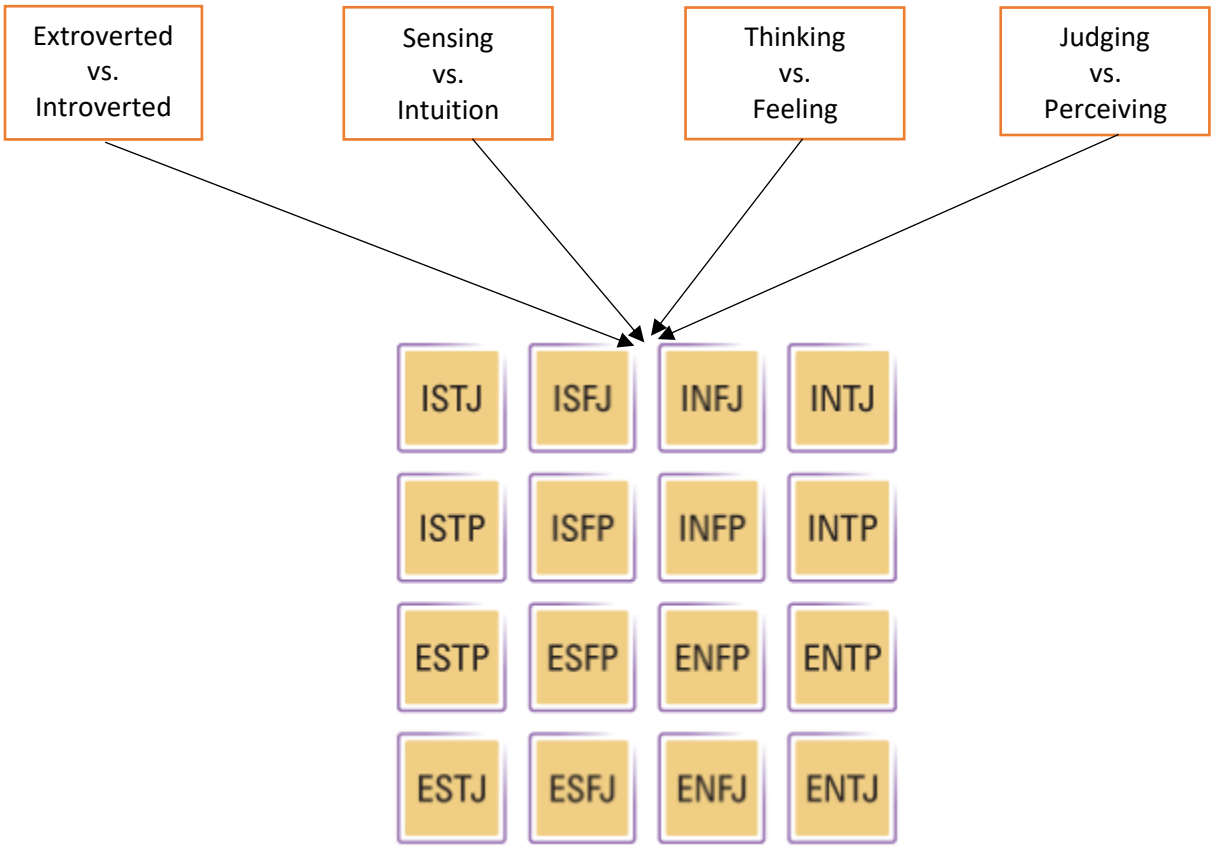

Figure 1. MBTI personality selection process [44]

The Sixteen personality types are described and explained [44]:

- **ISTJ**

  The people prefer to pay attention to physical factors in the real world such as specific standards lonely. The “Thinking” type indicates that people usually make decisions based on the objective truth instead of the feeling. And a planned or orderly life is usually conducted. So they are logical, detailed and organized people.

- **ISFJ**

When encountering the problems, they prefer to work alone based on the objective rubrics. But for making decisions, they would like to ask friends about what they care about instead of some basic principles. And for the outer life, they prefer to make a detailed plan instead of a flexible life. So they are quietly warm, realistic and organized people.

- **INFJ**

  They prefer to think/work independently or with only few friends. During solving problems, they pay attention to the patterns of information instead of some physical facts. Also, decisions are usually made based on what friends care about instead of their own thoughts. And a planned life is their preference options. So they are organized, detailed but emotional people.

- **INTJ**

  People would like to get the energy from dealing with memories and reactions in the inner world [44]. They prefer to pay more attention to the information's pattern instead of "spotted" information in their mind. Also, when analyzing the problems, the basic truth or principle is applied in their decisions. And they live with a planned life. So they have original minds and great drive for implementing their ideas and achieving their goals.

- **ISTP**

  They prefer to solve problems alone through analyzing advantages and disadvantages. Although they pay more attention to the physical realty based on what they touch, they live with a flexible style of life. So they are tolerant and flexible, quiet observers until a problem appears, then act quickly to find workable solutions [44].

- **ISFP**

  They solve problems alone with the objective standards and prefer to live a flexible life. But usually, decisions are made based on their friend. So they are quiet, friendly, sensitive, and kind and dislike disagreements and conflicts, do not force their opinions or values on others [44].

- **INFP**

  People utilize the pattern of information to solve problems alone and usually make decisions based on friends involved in the situation. For their outer world life, it is flexible. So they are idealistic, loyal and want an external life that is congruent with their values [44].

- **INTP**

  People pay attention to the pattern of information, try to solve problems independently and live a flexible life. So they seek to develop logical explanations for everything that interests them and are quiet, contained, flexible, and adaptable [44].

- **ESTP**

  People prefer to work with other friends with the physical reality. When making decisions, they usually applied the basic principles, regardless of the specific situation involved. So, they are flexible, tolerant [44] and take a pragmatic approach focused on immediate results.

- **ESFP**

  People would like to get energy from activities and to enjoy working with others. The decisions are usually made with other's opinions instead of their own judgment. For

their outer world life, it is flexible. They are outgoing, friendly, and accepting and to bring common sense and a realistic approach to their work [44].

- **ENFP**

  People enjoy working with others. They apply the pattern of information received from external sources (i.e. outside world) to solve problems. When decisions need to be made, they care about opinions from others evolved in a situation. And a flexible life is their favorite life. They are warmly enthusiastic, imaginative and see life as full of possibilities [44]. They make connections between events and information very quickly, and confidently proceed based on the patterns they see [44].

- **ENTP**

  They would like to communicate with others and to work together. For problems, they apply the pattern of information to think independently based on the principles instead of others' opinions. For the outer world life, it is flexible. Hence, they are quick, ingenious, stimulating, alert, and outspoken [44].

- **ESTJ**

  They are "extraversion" people and would like a flexible life. The style of solving problem is to use the principles, regardless of the situation involved. Based on the characters explanations, they are practical, realistic, matter-of-fact, decisive, quickly move to implement decisions [44]. And they have a clear set of logical standards, systematically follow them and want others to also. Forceful in implementing their plans [44].

- **ESFJ**

  People would like to work with others and make decisions with the consideration of others' suggestions. They like the planned or orderly life and usually pay attention to the physical reality instead of the abstract pattern of information. Hence, they are warmhearted, conscientious, and cooperative, want harmony in their environment and work with determination to establish it [44].

- **ENFJ**

  People would like to collaborate with others. They are boring at a planned life. But for solving problems, they apply the abstract pattern of information instead of the physical reality. Hence, they are warm, empathetic, responsive, and responsible and may act as catalysts for individual and group growth [44].

- **ENTJ**

  People enjoy the team work to solve problems and make decisions based on the true principles, regardless of the specific situation involved [44]. They are used to adopting the high-level type of thinking about problems and enjoy a planned life. Hence, they are frank, decisive, assume leadership readily, can quickly see illogical and inefficient procedures and policies and develop and implement comprehensive systems to solve organizational problems [44].

Many related studies on the MBTI personality in computer science were conducted to suggest what type of personality fits for which phase of the programming process. For example, Bishop [4] used MBTI personality model to evaluate and analyze the personality

type of college students in some programming courses and concluded which type of personality is best in each phase of programming work.

Based on the introduction of both the FFM model and the MBTI model above, we explain the reason that the MBTI model is selected in our research work instead of the FFM model. The Five-Factor personality model does not theorize about what goes on inside people's heads; it focuses on actual behaviors. But, the MBIT model characterizes people by their attitude towards the inner and outer world and is based on a cognitive theory that explains basic personality traits, as arising from differences in how we take in and process information. The theory of the MBIT model is what we use to understand the world. The Five-Factor model can only find correlations but not put them into any context – conservatives score high on Conscientiousness, liberals score high on Openness etc. Therefore, with the purpose of research work, we employed the MBIT model in measuring personality.

Although the personality is a psychological term and the programming performance is a metric in computer science, the relationship between the personality and the programming performance has been explored for several years. The first research on the potential interaction between two different disciplines was proposed by Weinberg [20] in 1971. The author explored the impact of diverse areas such as personality, motivation, intelligence, and experience on the various aspects of programming in groups. Also, the author suggested that different programming skills were required at each phase of the software development such as code review or code test, so few programmers can always make the best performance in

all phases, which is affected by their individual characteristics. Although this research did not present many statistical numbers to support his hypotheses, it guides us to conduct the relative experiments on the potential influence of the personality and programming activities.

Additionally, Zahra *et al.* [13] investigated personality's influence on the programming styles. But the use of FFM is less accurate, and the programming styles were collected between 1982 and 2006, which is quite out of date. What is worse, the programming styles were not validated. To improve this work, we utilized MBIT to measure the personality, and the contemporary programming styles are collected after 2006. The results are validated with Cronbach alpha.

Finally, we cannot ignore another important impact on the programming: the programming attitude. Psychologically, the attitude is defined as an overall evaluation of an object that is based on cognitive, affective and behavioral information [35]. In computer science fields, Eric *et al* [5] developed the programming attitude in which five factors were validated: (1) Confidence in learning computer science and programming; (2) Attitude toward success in computer science; (3) Computer science as a male domain; (4) Usefulness of computer science and programming; and (5) Effective motivation in computer science and programming.

To our best knowledge, there is no empirical study on the following two topics: the effects of the MBTI personality on programming attitudes, and the link between

programming styles and programming attitudes. In the present study, we make a first attempt to investigate and report the results of this empirical study.

1.2 General Research Area

Generally, the programming work is a human task, which is significantly affected by human factors such as the personality, programming styles, and programming attitudes. Although previous research work has investigated personality's influence on the programming styles, one problem for those investigations is that the study of programming styles in the paper did not keep pace with contemporary programmers' habits, such as parallel programming styles, the habits in a big team work and the way of rewriting the existing code. To extend the previous work, we collected the programming styles since 2006 and conducted an empirical analysis of personality and the contemporary programming styles.

In addition, programming attitudes also play an important role in programming work. In [21], authors developed a computer science attitude survey and collected 162 responses to validate the attitude survey. The results indicated that students with positive programming attitudes perform better on programming projects and are more likely to succeed by completing the class with a C or better. Therefore, we also add one more human factor--programming attitude--into our empirical investigation. Specifically, the following research questions are investigated: (1) the influence of personality based on MBTI model on programming styles; (2) the influence of personality based on MBTI model on programming attitudes; (3) the influence of programming attitudes on programming styles; (4) the

influence of personality based on MBTI model on the programming performance; (5) the influence of programming attitudes on the programming performance; (6) the influence of programming styles on the programming performance. To clarify our research work, the relationship among them is shown in Figure 2.

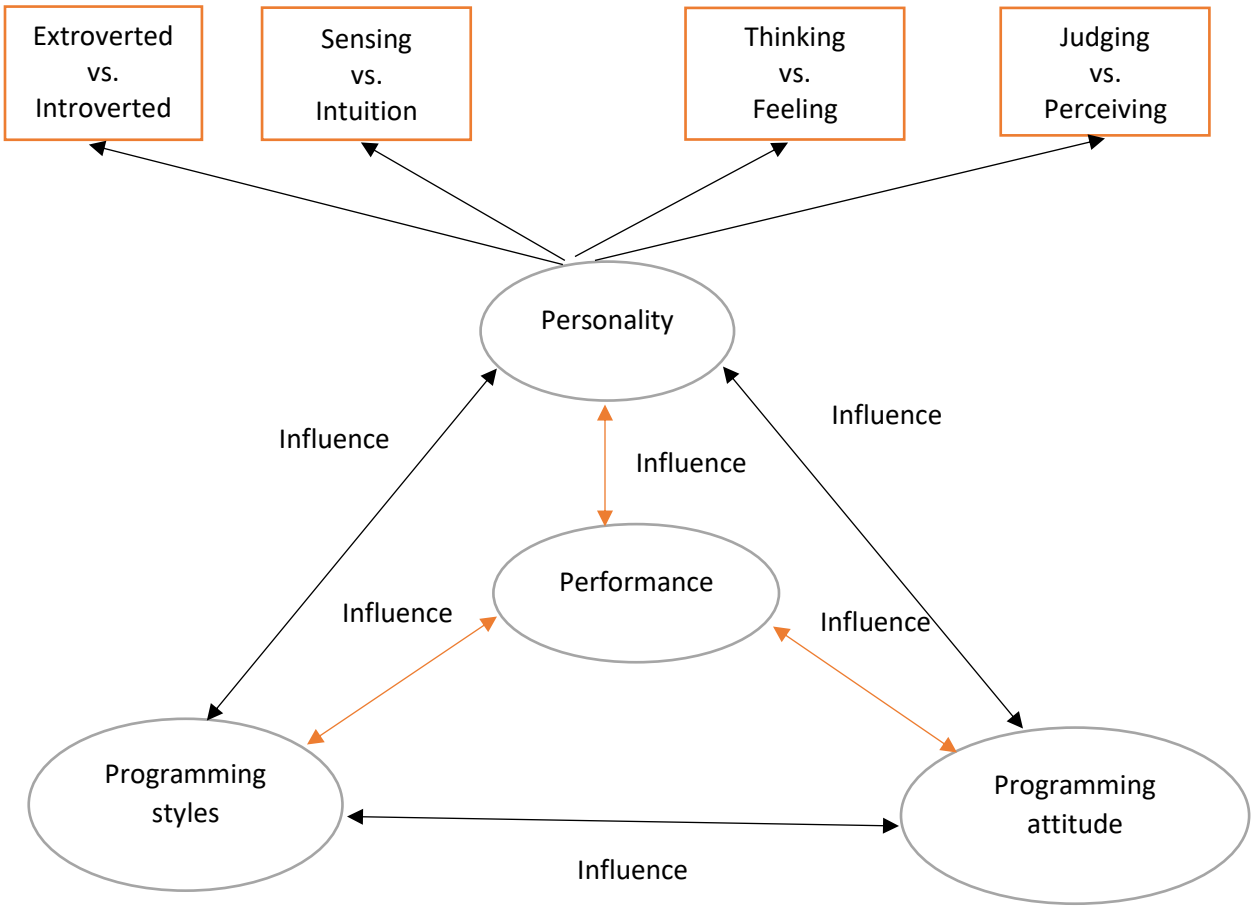

Figure 2. The research goals among personality, programming styles and programming attitudes

In order to complete programming work, many factors need to be seriously considered such as knowledge, programming skills, algorithms design, code synchronization and test,

and strategies in organizing the programming teams. Until now many research papers have investigated the relationship between the performance and human factors (i.e. personality, programming styles and programming attitudes). But researchers only used the test scores or the final grades to define the performance. Unfortunately, it is less convincing because code quality also needs to be considered seriously. For example, there could be the same grade for two code solutions of the same problem, but we cannot state they have the same code. Typically, a surprising difference usually exists between these two solutions such as code readability. To overcome this limitation, we enrich the previous performance definition with two factors: (1) running time of code; and (2) rubrics scores (subjectively evaluate the code).

1.3 Research Problem

From the perspective of technologies, the performance of programming models, especially for parallel models such as CUDA and OpenACC, has been studied for several years based on optimization. However, few papers explore the programming performance from the impact of human factors such as personality, programming styles, and programming attitudes. Some researchers did empirical studies about the influence of personality on the programming styles. For example, Zahra *et al.* [13,14] found that programming experience is the most influential factor in programming styles. Although Pearson correlational analysis was employed to make conclusions, the results were not precisely based on the following five limitations:

(1) The fact that programming performance was defined with final grades and exam scores makes their conclusions less convincing. More metrics need to enrich this definition.

(2) Although FFM has been widely used in the evaluation of the personality, this model does not theorize about what goes on inside people's heads; it focuses on actual behaviors instead of the theory such as the cognition theory.

(3) The survey of the personality using FFM [13,14] was conducted twice, but each time the questions were not consistent and the survey was not fully conducted.

(4) A similar problem happened to the survey of programming styles. Authors adopted ten questions of fifty-six original questions from five categories of programming styles. Although a few questions were selected from each category, the results were less convincing because the samples were less reliable.

(5) The programming styles were collected from research papers between 1985 and 2006, which are quite out of date. Before analyzing the relationship between the personality and programming styles, the author did not validate them.

Against those threats above, the FFM will be replaced by the MBTI model. And the evaluation of the personality and programming attitudes will be fully conducted in the department of computer science and software engineering. In addition, the programming styles will be updated to reflect styles that have been prevalent since 2006 and will assess the connection between personality and contemporary programming styles. We will preform a comprehensive exploration between the programming performance and human factors: personality, programming styles, and the programming attitude. We also will analyze the relationship among the personality, programming styles and the programming attitude with the expectation of guiding researchers in programming work.

1.4 Purpose of Research

The purpose of this study is to investigate the influence of human factors on the programming performance and to analyze the relationship among personality, programming styles, and programming attitudes. For programming styles, we will collect and validate the contemporary programming styles since 2006. With the data from full items of MBTI personality, we will explore which factor is the most influential for programming styles. Additionally, we will define the running time as the coding performance and the grades as the academic performance.

For programming styles and programming attitudes, we expect to find which attitudes are helpful in improving the programming styles and which styles need to be avoided in programming work. For the programming performance, we expect to identify: (1) positive human factors to help programmers improve their performance, and (2) negative human factors to prevent from performance deterioration. This exploration can be used to guide programmers in programming work. Therefore, we expect to show researchers: (1) which personality factors can positively improve the programming performance and styles; (2) which programming attitudes can positively improve the programming performance and styles; (3) which programming styles can positively improve the programming performance; (4) the relationship between personality and programming attitude.

1.5 Research Approach

This empirical study was conducted in three phases. In the first phase, the survey questions in the surveys of personality, programming styles and programming attitudes were confirmed. For the personality survey, the MBTI items were employed. For the programming styles survey, we collected the styles after 2006 from research papers, technique reports, books, and presentations. After the data collection of programming style survey, we validated them. The programming attitude survey was adopted from Eric's model [5].

In the second phase, students in the department of computer science and software engineering completed the self-assessed survey that was distributed to them in person or through an online survey that can be accessed at https://www.surveymonkey.com/r/ProgrammingStyle.

In the third phase, we used statistical methods such as Pearson Correlational Analysis, an independent sample T-test, and linear regression analysis to find the most influential factors.

1.6 Research Questions

Our empirical experiment aims to investigate (1) the relationship among personality, programming styles and programming attitude; and (2) effects of human factors on the programming performance. Since there are multiple aspects within each human personality category, we specifically analyzed the influence among each category with as many details as possible. Overall, the following questions will be investigated in this study:

1. Can contemporary programming styles be validated?

2. Which factors in a personality have positive/negative impacts on programming performance?

3. Which factor in programming attitude has positive/negative impacts on programming performance?

4. Which factor in programming styles has positive/negative impacts on programming performance?

5. What is the linear relationship based on Pearson correlation analysis among personality, programming attitudes, and programming styles?

1.7 Research Hypotheses

The MBTI personality model consists of four categories: (1) Where you focus on your attention – Extraversion (E) or Introversion (I); (2) The way you take in information – Sensing (S) or Intuition (N); (3) How you make decisions – Thinking (T) or Feeling (F); (4) How you deal with the world – Judging (J) or Perceiving (P). Under each category, one personality type indicator will be matched to a specific participant through self-assessment survey. Totally there are sixteen types: ISTJ, ISFJ, INFJ, INTJ, ISTP, ISFP, INFP, INTP, ESTP, ESFP, ENFP, ENTP, ESTJ, ESFJ, ENFJ, ENTJ. Based on the consideration of relationships between human factors and the performance, the following hypotheses are proposed in our study:

1. Since the Extroversion type indicates that people prefer to focus on the outer world, the programmers with this type of personality might like to write/debug the code in a group instead of working alone. And their solutions would be from different members in a group. So, the coding styles might not be consistent. The following hypotheses are proposed:

H_0: Extroversion indicator has a negative effect on the coding performance.

H_1: Extroversion indicator has a positive effect on the academic performance.

H_2: Extroversion indicator has a positive effect on the programming style of group coding in an office.

H_3: Extroversion indicator has a negatively linear dependence on the coding performance and academic performance.

2. Since the Introversion indicator indicates that people prefer to focus on their own inner world, the programmers with this type of personality might like to write/debug the code alone. Their code solution is consistent. The following hypotheses are proposed:

H_4: Introversion indicator has a positive effect on the coding performance and academic performance.

H_5: Introversion indicator has a positively linear dependence on the coding performance and academic performance.

H_6: Introversion indicator has a positive effect on the work-alone programming style.

3. Since the Thinking indicator indicates that people prefer to look at logic and consistency, from the perspective of programming context, programmers might write code with a text-based environment. The following hypotheses are proposed:

H_7: Thinking indicator has a positive effect and linear dependence on the coding performance and academic performance.

H_8: Thinking indicator has a positive effect on the programming style of the text-based programming environment.

H_9: Thinking indicator has a positive effect on the programming style of writing efficient code but is hard to understand.

4. Since the Feeling indicator indicates that people firstly prefer to look at the people and special circumstances, from the perspective of programming context, programmers might write code under the visual environment. The following hypotheses are proposed:

H_{10}: Feeling indicator has a negative effect and linear dependence on the coding performance and academic performance.

H_{11}: Feeling indicator has a positive effect on the programming style of the visual programming environment.

H_{12}: Feeling indicator has a positive effect on the programming style of writing redundant code, but is easy to understand.

5. Since the Intuition indicator indicates that people prefer to interpret information and add meaning, when testing programming, a programmer might do it unit by unit.

H_{13}: Intuition indicator has a negative effect on the coding performance and a positive effect on the coding performance.

H_{14}: Intuition indicator has a positive effect on the “unit-by-unit” programs style.

6. Since the Sensing indicator indicates that people prefer to focus on the structure of codes, a programmer might test all units at one time.

H_{15}: Sensing indicator has a positive effect on the academic performance and a negative impact on the coding performance.

H_{16}: Sensing indicator has a positive effect on the style of testing all units at one time

7. Since the Perceiving indicator indicates that people prefer to stay open to new options or information, programmers might like to synchronize their code with an automatic tool such as Github.

H_{17}: Thinking indicator has a positive effect and linear dependence on the coding performance and academic performance.

H_{18}: Perceiving indicator has a positive effect on the style of synchronizing project code automatically.

8. Since the Judging indicator indicates that people prefer to get things decided, programmers might like to synchronize their code manually without extra coding workload such as USB drivers.

H_{19}: Thinking indicator has a positive effect and linear dependence on the coding performance and academic performance.

H_{20}: Perceiving indicator has a positive effect on the style of synchronizing project code manually.

9. Positive confidence in learning computer science and programming has a positive role in: (1) the coding performance; (2) the academia performance; and (3) the collaboration style of developing software.

H_{21}: Positive confidence indicator has a positive effect and linear dependence on the coding performance and academic performance.

H_{22}: Confidence in learning computer science and programming affects the programmers' performance and collaboration styles of developing software.

10. The programmers with positive attitudes toward success usually prefer to interpret information instead of simply taking in basic information. Also, they prefer to stay open to new information and options.

H_{23}: There exists a positive relationship between the programming attitude toward success and the time continuity of programming.

H_{24}: Programming attitude toward success has a positive effect and linear dependence on the coding performance.

H_{25}: Programming attitude toward success has a positive effect and linear dependence on the academic performance.

H_{26}: A positive linear relationship between attitude toward success in computer science and Perceiving indicator exists.

H_{27}: A positive linear relationship between attitude toward success in computer science and Intuition indicator exists.

12. The programmers with a positive attitude of computer science as male domain might not affect the performance.

H_{28}: The strong linear dependence between performance and positive attitude of computer science as male domain does not exist.

13. The programmers with a positive attitude toward usefulness and effective motivation in programming might prefer to focus on the outer world instead of their inner world.

H_{29}: A positive relationship between attitude toward success and effective motivation in programming in computer science and performance exists.

H_{30}: Usefulness of computer science and programming has a positive effect on the Extroversion indicator.

H_{31}: Effective motivation in computer science and programming has a positive effect on the Extroversion indicator.

H_{32}: A positive relationship between attitude toward success and performance exists.

15. When facing bugs or testing code, programmers can easily concentrate on specific problems if the code is debugged or tested unit by unit, so debugging/testing code with the unit-by-unit style can efficiently improve the programming performance.

H_{33}: The “unit by unit” programming style of testing/debugging code plays a positive role in the programming performance.

16. In order to optimize the code, writing efficient code is compulsory. So naturally programmers with traits of writing efficient code but is hard to understand are highly possible to improve the programming performance.

H_{34}: The following programming style plays a positive role in improving programming performance: The programming style of creating thoughtful efficient code that increase program performance, as opposed to creating simple brute force solutions to programming problems.

1.8 Limitations

Although we try to analyze the influence of human factors on the programming performance as clearly as possible, some threats from the real world on this research still

exist. So, the first limitation is the educational level of participants. In this study, our participants are students at the undergraduate level. If experts or graduate students are recruited, the results may be different. In addition, since few students have parallel programming experience, we do not explore the programming styles of parallel computing. Thirdly, since the programming styles are a human programming behavior and are easily changed based on different types of population. We may continue to update items in programming styles several years later because new technologies will generate new programming styles.

1.9 Key Terms

Five Factor Personality model – Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism.

Myers-Briggs Type Indicator model – Introversion VS Extraversion, Sensing VS Intuition, Thinking VS Feeling, Judging VS Perceiving.

Programming Style - A term used to describe the effort a programmer should take to make his or her code easy to read and easy to understand (McCann, 1997).

Programming Attitude - attitudes towards computer programming and computer science in general.

Chapter 2 Literature Review

2.1 The related work in programming styles

The quality, efficiency, and performance of the code is highly dependent on the users' programming strategies. For example, some programmers prefer to divide the project into several sections and then to code each section, while some programmers would like to directly write the code to finish the whole project. Cox and Fisher [3] used the term "Programming Style" to describe these kinds of programming strategies for manipulating source code. In the paper of Cox *et al.* [3], the following conclusion has been made: experts or experienced programmers prefer to use systematic (breadth-first) to debug source code while novices focus on the specific(depth-first) line code to debug source code.

On the other hand, programming is a human task and different programmers work to code with different programming styles. In order to identify the human differences, we used the psychological term—personality—to distinguish behaviors, emotions, and cognitions among programmers. In the following section, the corresponding research work on programming styles is reviewed.

Iris [1] conducted an exploratory study to investigate the different programming strategies in an expert-novice team. All participants were required to debug pieces of Cobol code and their voice was recorded and then transferred to the identification number in the paper. The author found that experts like to use a systematic approach (breadth-first) to debug, while novices used

the depth-first approach to debug programs. There is a more efficient way to debug the source code when the participants used a systematic approach to debugging.

Cox and Fisher [3] developed a contextual, model-independent framework in which tasks, situation, and individual traits were combined. According to the components of this framework, educators can better understand how programmers produce source code. In addition, authors documented some components of programming styles. For example, some programmers like to work on a preferred amount of code for a preferred length of time.

Andrew and Bob [2] investigated the effect of individual differences on the program comprehension strategies under an unfamiliar debugging environment. To assess individual differences, the authors mainly tested the verbal intelligence, general problem solving ability, domain knowledge and basic information from the background questionnaire. Participants were required to debug two tasks under Intercooled Stata 7.0 environment. Results showed that individuals with stronger domain knowledge for specific bugs tended to be successful.

2.2 The related work in programming attitudes

Another important factor affecting programming is computing attitudes. Allison *et al* [4] hired 447 introductory students to develop a newly designed computing attitudes survey in 2011. Through interviewing 11 faculties and 9 students from both research-intensive and teaching-intensive institutions, this survey was distributed to participants and validated statistically. In order to categorize responses to individual questions, an exploratory factor analysis was

conducted. Finally, eight candidate factors were identified in this survey. Another computer science attitude survey was developed by Eric *et al.* [5] in 2003. This survey is mainly for measuring the attitudes of computer programming and computer science in general. There are five survey subscales and each subscale consists of a series of positive and negative statements. The reliability of the survey was evaluated with the responses from 162 students based on the internal consistency of these five subscales. Finally, this survey was validated with the Cronbach alpha in the 0.7 level.

Laurie *et al.* [21] conducted a formal paired-programming experiment at North Carolina University in 2001. In this empirical experiment, the authors compared the performance of exams and of programming projects. The results showed that paired student are more self-sufficient which reduces their reliance on the teaching staff. Qualitatively, paired students demonstrated higher-order thinking skills than students who worked alone. In addition, this computer science attitude survey was also validated with Cronbach's Alpha value in the experiment.

To better understand the fact that the enrollment in computer science was declining, authors [37] proposed a valid and reliable survey to examine science and engineering students' attitudes toward computer science. The participants were undergraduate students from Colorado School of Mines and the five constructs in this survey were identified and validated by Cronbach's Alpha. The results showed that this instrument can accurately measured the five constructs that need to be assessed.

2.3 The related work in personality theory

The Myers-Briggs Type Indicator, developed to make C. G. Jung's personality type theory understandable and useful in people's lives [22,23], has become the most trusted and widely-used psychometric instruments for assessing personality characteristics in non-psychiatric populations. Researchers have proved the validity and reliability of the MBIT.

For reliability, early literature reviews [24,25] based on the first version of the MBTI manual [26] gave: (1) internal consistency reliabilities for the four scales ranging from 0.75 - 0.85 with a low coefficient of 0.44 for the TF index, and (2) test-retest correlations of about 0.70 for three of the scales and 0.48 for the TF index. They found these statistics were comparable to the leading personality inventories at the time, but stated the need for more reliability studies. Harvey [27] evaluated and summarized results of research on the MBTI's reliability and validity in the ten years following the publication of the manual in the second edition. Results of meta-analytic studies, using generally accepted standards applied to instruments with continuous scores, show that reliabilities of the MBTI continuous scores were quite good—average overall reliabilities of 0.84 and 0.86 for internal consistency measures, and 0.76 for temporal stability.

For validity, Myers *et al.* [22] noted that the correlations could be expected to underestimate the magnitude of the relationships. Correlations with the Jungian Type Survey (JTS) indicates a significant commonality of constructs being tapped by both, though with more consistency for the EI and SN scales than the TF (E = 0.68, I = 0.66, S = 0.54, N = 0.47, T = 0.33, and F = 0.23).

Harvey in [27] summarized the expansion of validation research and increasing empirical evidence in support of the MBTI's convergent, divergent, and predictive qualities in the recent decade. Fleenor in [28], in his critical review of Form M, concludes that Form M has significantly improved the MBTI, citing improved scoring procedures with the use of IRT, and eliminating of gender differences in some scales using DIF analysis.

An alternative measurement Five-Factor Model (FFM) for distinguishing individual differences--the measurement of the personality--is the most commonly used method in the academia. Normally, researchers would like to evaluate individual personality with Five-Factor Model [6] consisting of Openness to Experience, Conscientiousness, Extraversion, Agreeableness and Emotional Stability. Schmit and Ryan [7] analyzed the responses to individual items of a short form of NEO-PI (Personality Inventory) within applicant and non-applicant samples. The results showed that an exploratory analysis (EFA) of a non-applicant sample demonstrated the expected five-factor solution.

In addition, some researchers tried to add other methods to improve the Five-Factor model. Cellar *et al.* [8] used the confirmatory factor analysis to estimate a sixth factor and they found that adding the sixth factor significantly improves NEO-PI model. Similarly, Lim and Ployhart [9] used a multi-trait, multi-method (MTMM) analysis of scale scores to find that adding two orthogonal factors, one associated with NEO_FFI data and one associated with International Personality Item Pool (IPIP), improved models.

As discussed above, two of the most popular personality assessments are NEO-FFI and the IPIP scales. Although the NEO personality inventory developed by Costa *et al.* [10-11] is the most famous one, this is the commercial inventory, so researchers prefer to use IPIP (International personality Item Pool) to conduct the relative experiments. In paper [12], the structure of 50-item IPIP was validated in three different adult samples (N=906). The data from 3 questionnaires showed that there is a high internal consistency in the IPIP model.

The statistical data in paper [36] state that until 2015, (1) 40% of studies used MBTI, and (2) 23% of empirical studies used FFM. Figure 3 indicates a higher increase happened to FFM model compared to MBTI model since 2000.

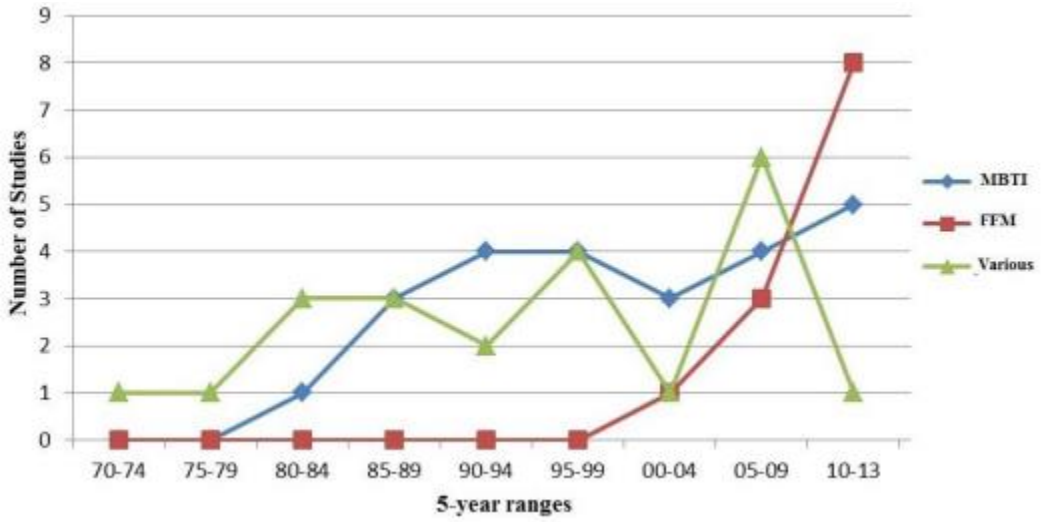

Figure 3. The trend of personality tests used in studies on personality and computer programming

2.4 The related work in links between programming styles and personality

Although there is various research work on the personality assessment and the programming styles, few papers focused on the research of influence of personality on programming styles. Zahra *et al.* [13-14] hired 68 students at University of Stuttgart to finish a self-assessed survey on programming experience, five factor personality, attitude towards programming and programming style. To validate their questionnaires, authors conducted the piloted study twice before the true survey distribution was conducted among students. With Pearson correlational analysis, regression analysis and mean analysis, the findings showed that programming experience was the most influential factor in programming styles. In addition, among components of the Five-Factor model, authors also found a positive relation between Openness to experience and the systematic programming styles. Finally, the conclusion was made that conscientiousness is the most influential personality factor on programming styles.

2.5 The related work in links between team/group/pair programming and personality

Mark [33] did a comprehensive investigation on the effects of group personality composition on project team performance. The hierarchical regression was employed in the statistical analysis of nine hypotheses. In his dissertation, the author separately analyzed the relationship between team performance and five group personality compositions: Team Conscientiousness, Team Extraversion, Team Emotional Stability, Team Openness to Experience and Team Agreeableness. The results showed that the group personality composition of a team significantly affected its performance and that each specific group personality trait predicted team performance.

John *et al.* [34] developed a personality model of the Myers-Briggs Type Indicator (MBTI) to predict the team performance. In this model, the four factors (Leadership, Cohesion, Communication and Heterogeneity) were established and a modern information system (IS) was used to test the validity of this model between two teams. The results from the analysis of Critical thinking, IQ, Age and the components of MBIT showed that this model is useful to predict the team performance.

Pair programming is a practice, whereby two programmers work side by side at the same computer, continuously collaborating on the same design, algorithm, code, or test [31]. One of the programmers has the control of the keyboard and the mouse, actively implements the program, and explains the implementation to his or her partner. Studies indicated that students with pair programming style in introductory computer science courses had an equal or higher chance of passing the course with a C or higher, produced better programs, and were more likely to pursue the computer science major than students who solo programmed [32].

Another research [29] was conducted among a pilot study of 34 introductory programming classes at a medium-size Midwestern university and a follow-up study of 114 college students attended a different campus of the university, taking the same computer programming class as the pilot study students. This research indicated that there is a relationship between programming performances and the four MBTI personality dimensions.

David Keirsey [30] conjectured that different types of people have different sets of strengths and weaknesses. The strengths and weaknesses of different types complement each other. They contended that if the students with different personality types are given an opportunity to work together, they could produce better results based on different opinions.

2.6 The related work in links between performance and personality

Pressman [40] 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.

Amy *et al.* [41] noticed the fact that learning how to program is difficult, and failure and attrition rates in college level programming classes remain at an unacceptably high rate. Although many educators simply accept this high failure rate, authors tended to explain this phenomenon from cognitive profile. Krause's cognitive profiles were distributed to 246 students at Southeast University. Through ANOVA analysis, authors recommend instructional strategies that may be used to reach fully motivated and intellectually capable sensor feeler. Eventually, Krause's Cognitive Profile Inventory (CPI) classifies people on the areas from Myers-Briggs, so the CPI provides a valid and shorter alternative to Myer-Briggs.

In [42] authors develop a model for assessing personality traits in Software Engineering(SE). An assessment technique based on Big Five Factor model and Myer-Briggs model was designed to guide intending software engineers in choosing specialization areas based on their personality traits. Finally, 58 seniors were recruited to evaluate this model. Results showed that the current model can assist first-year student to make a correct decision in the selection of specific research direction.

Similarly, Cunha *et al.* [15] investigated whether or not the programming performance is linked to the personality type under the Myers Briggs Type Indicator (MBTI) models [16-19]. Participants were required to find the bugs in the software (Java code). In order to analyze the results statistically, the various bugs were weighted according to the difficulty. The data from Pearson correlations showed that participants with thinking and sensing personality are best at solving problems so Weinberg [20] hypothesis was validated.

2.7 Distinction of My Research

This study enriches factors in defining programming performance: coding performance (running time) and academic performance (grades) and is the first time to measure the run time in analyzing the influence of human factors on performance. Secondly, although researchers had investigated the influence of personality on the programming styles, all programming styles were collected from 1985 to 2006. With the development of programming languages such as the appearance of parallel models, a dramatic change has happened to programming styles. In our empirical study, we collected programming styles after 2006 and plan to validate them. Another

contribution is that we specifically investigate the effects of each factor of personality using the MBIT model on programming styles and performance. Also, computer science attitudes are investigated with the consideration from personality. Finally, we also add the programming experience and the programming attitude to our experiment. Therefore, we will present guidelines for educators in the programming field based on this comprehensive investigation.

Chapter 3 Method

3.1 Participants

In our study, the participants with programming background are expected because we investigate the programming performance under the impact of programmers' personality, programming styles and attitudes. In the initial phase, the students in the department of computer science are the target of the experiment. In the future work, we will invite experienced programmers and experts who have worked in programming for, at least, ten years to enrich our data.

3.2 Subjects description

Software development is a human task and the individual differences result in different programming styles. Although researchers did excellent work on extracting the programming styles such as Vessey's exploratory study of debugging code in 1982 and Cox's contextual framework in programming styles in 2005, few studies work on contemporary programming styles such as parallel programming styles, programming style in the team work and so on. To fill this gap, we update the programming styles after 2006 and also investigate the influence of contemporary programming styles on the programming performance.

For personality, the big Five-Factor model and Myers-Briggs model are mainstreams in currently academic research and career evaluation. Since the big Five-Factor model does not theorize people's behavior, the Myers-Briggs model is naturally adopted in our research. The personality is fully evaluated with 48 questions of four main factors: Extroversion vs.

Introversion, Sensing vs. Intuition, Thinking vs. Feeling, and Judging vs. Perceiving. For each question, participants will check one of the five options: strongly agree, little agree, neutral, little disagree, strongly disagree.

To enrich items of programming styles, we add contemporary styles to the new programming habits. For example, currently hardware is updated faster and faster, which means it can afford more computing or bigger projects. So, the "group programming" style seems to become the mainstream in developing software. Based on the group programming styles, we are interested in the responses from participants. Besides, the programming styles on debugging/testing and the programming workplace are also investigated.

For the definition of programming performance, we define the running time of code as coding performance and programming scores based on rubrics as the academic performance to evaluate programmers' performance as objectively as possible.

3.3 Instrumentation

For data collection, we plan to distribute three questionnaires in the classroom or online. Forty-eight questions related to the Myers-Briggs Type Indicator personality, a survey of computer science attitude and the new questionnaire on programming styles are created to collect data. To verify whether participants have programming experience in some projects, background data are added as a pre-questionnaire. In the analysis phase, the correlation methods such as

Pearson correlation analysis are mainly employed to explore the relationship between each factor of personality and programming styles/the performance. And we also use the regression model to predict the impact of human factors on the performance. Finally, the independent T-test is employed to find whether there exists the significant performance difference among dimensions of human factors.

3.4 Procedure

To perform the experimental tasks, participants might have access to the Internet through a web browser at https://www.surveymonkey.com/r/ProgrammingStyle. The questionnaires can also be accessed through hand-held devices, iPad, smart phones, laptops and desktops. For participants who are not able to login online, we will distribute a questionnaire of paper version to them.

Based on the feedback from participants, we will utilize statistical analysis such as Cronbach Alpha to validate the survey of contemporary programming styles, which consists of 9 categories. We will use the survey categories whose Cronbach Alpha is greater than 0.7 in our research work. After the data collection, all responses will be protected by computers with password protection or stored in a locked room.

3.5 Data Collection

The data collection of the research experiment is mainly based on the Internet. The subjects who have agreed to participate will be given a link to a website with the details of the study in order for them to become familiar with the study at their own convenience. Before taking part in

the study, participants will be provided an IRB form to explain their rights. An Information Letter states that participation is purely voluntary, and that they can withdraw from the study at any time without any given reason. The experiment process will have five sections (Figure 4) that includes IRB forms, the completion of survey forms by participants via Internet or paper forms, the data stored and the data process. Approximately ninety-five questions will be provided.

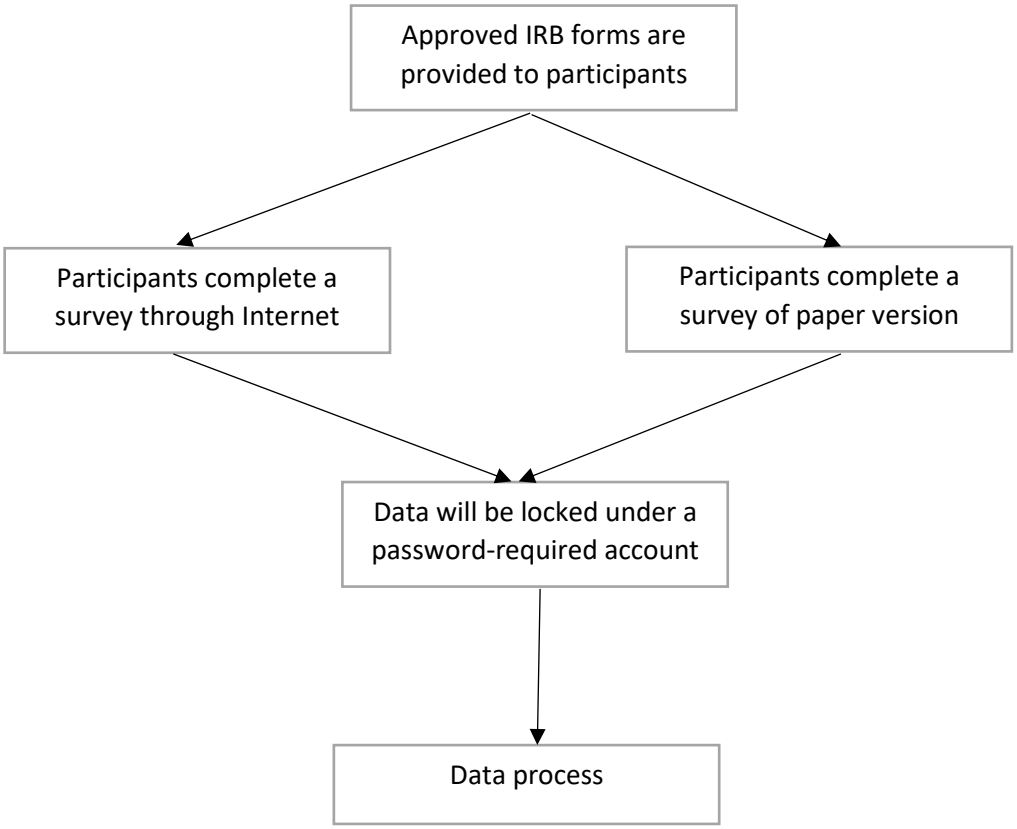

Figure 4. The experimental process

3.6 Data analysis

We updated the original programming styles since 2006, so before analyzing the links among programming styles, personality and programming attitudes, the programming styles need to be validated. Because the Cronbach's Alpha is a measure of internal consistency, that is, how closely related a set of items are as a group, it is considered to be a measure of scale reliability. Statistically, we adopted a 0.7 level to accept the dimension of programming styles. That is, if the Cronbach Alpha value of one dimension is greater than or equal to 0.7, we will use it in our analysis research work; if the Cronbach alpha value of one dimension is less than 0.7, we will remove this dimension from the analysis research work.

Also, we analyzed the influence of human factors on the programming performance and the links among programming styles, personality and programming attitudes. We use Pearson Correlation Analysis and Linear Regression Analysis to make conclusions in our study because: (1) Pearson Correlation Analysis is a measure of the strength of a linear association between two variables, and (2) Linear Regression Analysis is used to describe data and to explain the relationship between the dependent variables and the independent variables.

3.7 Experimental phases

In the experiment, we conducted three phases of assessments through questionnaires. In phase one, we preformed pre-design work such as requirement analysis and conceptual designs. To identify the items of programming styles, the researcher proposed a series of face-to-face interview with programming professors and performed a literature review on the topic of programming styles. The objectives in phase one are to:

1. Identify the contemporary programming styles published by researchers.
2. Identify the contemporary programming styles after 2006. See Appendix C for a list of programming styles.
3. Identify the available personality inventory. See Appendix A for a list of personality.
4. Identify the available programming attitudes. See Appendix B for a list of programming attitudes.
5. Gather information and conduct the proper questionnaires for our research.
6. Based on information of personality and of programming styles, prepare for the form of research involving human subjects.

In phase two, participants voluntarily attend the survey recruitment. The main activities include:

- The project investigator will explain the purpose of the experiment and relevant instructions, and show participants the approved the form of research involving human subjects.
- The link of the survey will be provided. For participants who prefer to fill out the survey of a paper version, we can also distribute the surveys to students in the classroom.
- The survey tasks will be performed by participants at any time and at any place through Internet.
- The data will be saved in the password-protected computers or the locked room.

In phase three, all data will be compiled and processed based on statistical metrics:

- Cronbach's Alpha will be employed to validate the programming styles survey.

- Independent sample T-test, Pearson Correlation Analysis (PCA) and Linear Regression Analysis (LRA) is used to analyze the influence of human factors on the programming performance and the links among personality, programming styles, and programming attitudes.

Chapter 4 Comprehensive Evaluation

In this chapter, the empirical results are presented and discussed. And the conclusions are made based on experimental data and statistical analysis. The terminologies and background in the personality, programming attitudes, and programming styles are explained in details in section 4.1, 4.2 and 4.3. Also, the skeleton of three surveys are associated with the terminologies explanation. Secondly, because there are several categories of each survey in our empirical research, it is necessary to discuss the motivation in selecting statistical analysis methods when we analyzed the relationship between the programming performance and human factors. In section 4.5, it is presented that all empirical results and conclusions with statistical standards. For example, the value 0.05 was used as a threshold to make conclusions whether or not the comparative samples are significantly different.

4.1 Empirical studies in the personality survey

Since the Myers-Briggs model is adopted in our research, four categories under this model are introduced based on the Myers & Briggs Foundation [44]. Table 2 illustrates each category, index of measurement, purposes of measurement, and corresponding questions used in the research. In the "Survey Questions" column, the question number corresponds to the order of Appendix A: Personality Questionnaire.

Table 2. Myer-Briggs Personality basic concepts

Category	Index	Purpose	Survey Questions
Favorite world	Extraversion(E) or Introversion(I)	Participants prefer to focus on the outer world or on	E: Q1-Q5 I: Q6-Q10

		their own inner world or not	
Information	Sensing (S) or Intuition (N)	Participants prefer to focus on the basic information they take in or prefer to interpret and add meaning	S: Q11-Q16 N: Q17-Q22
Decision	Thinking (T) or Feeling (F)	When making decisions, participants prefer to first look at logic and consistency or first look at the people and special circumstances	T: Q23-Q29 F: Q30-Q36
Structure	Judging (J) or Perceiving (P)	In dealing with the outside world, participants prefer to get things decided or do you prefer to stay open to new information and options	J: Q37-Q42 P: Q43-Q48

In order to classify the personality of each student, the data processing was described in the following steps. Since there are five potential options for each participant, the weight is relatively associated with each option. The weighted-option table is described in Table 3.

Table 3. Weight-Option values

Options	A: Disagree	B: Little Disagree	C: Neutral	D: Little Agree	E: Agree
Weights	1	2	3	4	5

The survey answers can be numerated between 1 and 5. Each category consists of a pair of preferences which are opposite. For example, there are two preferences in Decision category: Thinking (T) and Feeling (F). The Thinking preference is completely opposite to the Feeling preference. We use Thinking as the first preference and Feeling as the second preference.

For all data in Decision category, we made a summation in the first preference. For the second preference, all questions are used to measure participants' opposite preferences so the results can be obtained with a calculation of subtracting the points by total points. For example, the Feeling preference of the Decision category has 7 questions and total points are 35 by the 1-5 point (i.e. Likert scale). In order to process the valuation for a student's personality between Thinking and Feeling, there is a maximum of 35 points that can be attributed. If a student scores 31 points in Thinking(T) preference and 21 points in the Feeling(F) category, we feel that this is not identified as strongly Thinking. In order to attribute the proper amount of T/F preference, we record both values and calculate the T/F measurement. We calculate T/F by storing 35 then subtract 21 and the calculate T/F points for this student in Feeling preference is 14. And the last step is that we added the score (14 points) from opposite section to the score from the first section. Finally, the points in Decision category for this student is the value of 31 plus 14: 45. We define this conversion from the second preference to the first preference the normalization and applied it in all participants and all dimension of human factors.

Until now we have normalized all scores for each category and then the mean value of the scores under this category is used as a middle point to classify participants' personality. Taking the calculation of the "Decision" category as an example, the researcher assumes that the mean value is 35 among 320 results. Since the score is 44 for this student, it is classified as Thinking type. If the mean value is 45, this student will be classified as Feeling type.

4.2 Empirical studies in the programming attitude survey

In measuring programming attitudes, the questionnaire was adapted from Eric [5], which was composed of five categories. Table 4 summarizes different categories, survey questions and the purposes of measuring each category. And the survey questions are indexed according to the order of Appendix B: Programming Attitude Questionnaire.

Table 4. Programming attitude basic concepts

Category	Index	Purpose	Survey Questions
Confidence	Confidence(C) or Non-confidence(NC)	Confidence in learning computer science and programming	C: Q1-Q6 NC: Q7-Q12
Success Attitude	Success Attitude(SA) or Unsuccess Attitude(UA)	Attitude toward success in computer science	SA: Q13-Q18 UA: Q19-Q24
Gender domain	Male Domain(MD) or Female Domain(FD)	Computer science as a male domain	MD: Q25-Q28 FD: Q29-Q32
Usefulness	Usefulness(U) or Unusefulness(UN)	Usefulness of computer science and programming	U: Q33-Q38 UN: Q39-Q44
Effectiveness	Effectiveness(E) or Ineffectiveness(IE)	Effective motivation in computer science and programming	E: Q45-Q49 IE: Q50-Q56

To normalize the survey results of programming attitudes from students, the data processing method is the same as the calculation in the personality survey. Also, the weight of each option for every question is the same value as shown in Table 3.

4.3 Empirical studies in programming styles survey

Since more and more researchers and educators noticed that the performance of the source code and programmers' efficiency is highly dependent on programming habits, which are called the programming styles in our research. Hence, we collected and developed a contemporary programming style survey since 2006 with the hope of assisting programmers, especially for

novices, in improving the performance and programming efficiency. As Appendix C shows, we investigated the participants' academic background and totally there are 23 questions under 9 categories. Details of programming styles are illustrated in Table 5.

Table 5. Programming styles basic concepts

Category	Index	Purpose	Survey Questions
Programming Collaboration	Programming Alone(PA) or Programming in a Group(PG)	Prefer to work alone or Prefer to work with others	PA: Q7, Q10, Q11 PG: Q8, Q9, Q12, Q13
Programming Time Duration	Continuous Programming(CP) or Intermittent Programming(IP)	Program consecutively or Program intermittently	CP: Q14 IP: Q15
Software Maintenance	Open-Source(OS) or Closed-Source(CS)	Software maintenance in the way of open-source or closed-source	OS: Q16 CS: Q17
Programming Context	Visual Programming(VP) or Text Programming(TP)	Programming in a visual context or in a text context	VP: Q18 TP: Q19
Programming Debugging	Partially Test(PT) or All-Block Test(AT)	Code debugging unit by unit or all-unit test at a time	PT: Q20 AT: Q21
Programming Environment	Programming Remotely(PR) or Programming Together(PT)	For programming, physically sit together to contact or contact in a remote way	PR: Q22a PT: Q22b
Software Requirement Analysis	Concrete Discussion(CD) or Abstract Discussion(AD)	Requirement discussion with a concrete picture or in an abstract way	CD: Q22c AD: Q22d
Code Understanding and Efficiency	Efficient-But-Confused Code(EC) or Redundant-But-Clear Code(RC)	Efficient code but hard to understand or redundant code but easy to understand	EC: Q22e RC: Q22f
Source Code Synchronization	Manual Synchronization(MS)	Synchronize group members' code	MS: Q22g

	or Automatic Synchronization(AS)	manually or automatically	AS: Q22h

To normalize the results of the programming style survey from students, the data processing method is the same as the calculation in the personality survey. Also, the weight of each option for every question is the same value as shown in Table 3.

4.4 Experiment Configuration

In Table 6, we list the demographic information of participants in our empirical experiment. Totally there were 421 participants, but 324 of them voluntarily participated in this research work. In Table 6 we can see: (1) the age of most participants was 18 years old or 19; (2) 257 of 324 (79.3%) participant was male; (3) 74.3% participants only have 0-6 months programming experience; (4) most students have the programming experience of less than 1000 lines of code; and (5) overall academic achievement is B+ level.

Table. 6 Participants demographics

Biographical Variable	Item	Number of Participants
Age	17	2
	18	118
	19	113
	20	54
	21	22
	22	7
	23	4
	24	1
	25	2
Gender	Male	257
	Female	64
	0 month	6
	0 month—6 months	241

Year of Programming Experience	6 month—12 months	16
	>12 months	43
Programming Language	C/C++	37
	Java	31
	C#	4
	Python	15
	CUDA	1
Largest Code Contribution	<100 LOC	190
	100 LOC – 1000 LOC	122
	1000 LOC – 5000 LOC	12
Academic Achievement	Average (B or above)	229
	Above Average (A)	93

Additionally, we briefly describe 7 projects used in the experiment. In Table 7, we can see that totally there are 4 assignments: (1) two simple projects are included in the first assignment; (2) there is only one project in assignment 2 which requires students to optimize the bubble sort algorithm; (3) the fourth project is included in assignment 3 which needs to implement a program to operate data from .txt files; (4) three projects are included in assignment 4, which is the hardest part of the experiment.

Table 7. Programming problem description

Assignment	Project	Description
A1	1	Solve a basic arithmetic statement: -3*3/5+13^2-(4*70)/(33*14)
	2	Use the law of sines to calculate the length of the missing side c of a triangle Delta with built-in Matlab functions, basic arithmetic and algebra.
A2	3	Implement an algorithm which improves bubble sort algorithm with a specific pseudo-code.
A3	4	Write a program that will populate a file with an NxM matrix, then it will read the NxM matrix in from the file and perform some basic mathematical operations on it (calculate the mode, avg, and find the min, max, and median in the matrix).
A4	5	Write a program that stores all grades and weights in a text file called "grades.txt" to calculate the percentage based on the grade with the following two function: (1) process_file; and (2) calc_grade
	6	Write a program that is able to read in expressions and values from a file then solve them using the function you created. The file is constructed: (1) The first line of the input file will tell you how many expressions are in the file; (2) Each subsequent pair of lines will consist of coefficients for x, y, and z; (3) the actual values you should substitute for x, y, and z

	7	Write a program that is able to read in equations from a file and then solve them using the **solve** function. Assuming solving linear equations with 3 unknowns. The file is constructed: (1) The first line of the input file will tell you how many triplets of expressions are in the file;

Since there are 7 projects tested in the research, we need to convert the data of all projects into a normalization value to predict the impact of human factors on the performance. The following steps are described in order to normalize the data. This process is listed in the following steps:

(1) For each project, find the maximum and minimum numbers;

(2) Convert the number to a standardization value with the formula: 0.1+0.8*(x- min)/(max-min) (x is the specific students' score);

(3) Mathematically calculate the average number for each student.

4.5 Influence of human factors on the programming performance and quality

Currently, the most evaluation of the code performance is conducted based on the running time. But other details of code writing are ignored such as the conciseness of codes, the clean of codes, the readability of codes and so on. Therefore, we develop the rubrics of assessing "non-running-time" factors before participants' projects were graded. The running time of codes is used to evaluate code performance. And grades of projects are used to evaluate academic performance. To achieve the evaluation statistically, the independent T-test is applied in exploring significant differences with a standard level of 0.05. The analysis of Pearson correlation and linear regression are employed in predicting effects of the human factor in performance. The R-square of 0.7 in linear regression is typically used to determine whether the prediction is convincing or not. And the value of 0.7 in Pearson correlation analysis is used to

make conclusions on linear correlation relationship among performance and human factors. Finally, novel programming styles were developed and verified with Cronbach Alpha. Statistically, if the value of Cronbach Alpha for each subscale in programming styles is greater than 0.7, we believe this subscale is verified. Otherwise, if the value is less than 0.7, this subscale would not be accepted.

Since three human factors (i.e. personality, programming attitude and programming styles) are investigated in their impact on programming performance, the following relationships are comprehensively explored, discussed and analyzed.

(1) which factors of personality significantly play a positive/negative role in the coding performance;

(2) which factors of personality significantly play a positive/negative role in the academic performance;

(3) which factors of programming attitude significantly play a positive/negative role in the coding performance;

(4) which factors of programming attitude significantly play a positive/negative role in the academic performance;

(5) Whether the programming styles can be verified;

(6) which factors of programming styles significantly play a positive/negative role in the coding performance;

(7) which factors of programming styles significantly play a positive/negative role in the academic performance;

(8) whether there exists a strong linear relationship among factors of personality and the code performance or not;

(9) whether there exists a strong linear relationship among factors of personality and the academic performance or not;

(10) whether there exists a strong linear relationship among factors of programming attitude and the code performance or not;

(11) whether there exists a strong linear relationship among factors of programming attitude and the academic performance or not;

(12) whether there exists a strong linear relationship among factors of programming styles and the code performance or not;

(13) whether there exists a strong linear relationship among factors of programming styles and the academic performance or not;

(14) whether the impact of personality factors on the code performance can be predicted or not;

(15) whether the impact of personality factors on the academic performance can be predicted or not;

(16) whether the impact of factors of programming attitude on the code performance can be predicted or not;

(17) whether the impact of factors of programming attitude on the academic performance can be predicted or not;

(18) whether the impact of factors of programming styles on the code performance can be predicted or not;

(19) whether the impact of factors of programming styles on the academic performance can be predicted or not;

(20) whether there exists a strong linear relationship between the personality and the programming styles or not;

(21) whether there exists a strong linear relationship between the programming attitudes and the programming styles or not;

(22) whether there exists a strong linear relationship between the personality and the programming attitudes or not.

4.5.1 Influence of personality on performance

Since there are four subscales of the personality, we respectively analyzed the impact of each subscale on the performance. And the performance under each subscale also was compared with the p-value metric of the independent sample T-test. In Table 8, the data illustrates the coding performance difference under each subscale with 7 projects.

For “Favorite World” category, the participants with Introversion characteristic can write more efficient code than participants with Extraversion characteristic except Project 4. In Project 4, although the running time of code written by Introversion students is slightly longer than one of code written by Extraversion students, the P-value 0.3826 indicates that there is no significant difference. Hence, statistically the coding performance of introversion is not worse than one of

extraversion in Project 4. We made a conclusion that generally, the introversion programmers outperform the extraversion programmers.

For “Information” category, since Intuition people pay the most attention to impressions or the meaning and patterns of the information rather than the physical reality in Sensing, we propose that the Intuition students would write more efficient code than Sensing students. All testing data from 7 projects support our hypothesis. Under each project, the running time in Intuition is significantly shorter than the running time in Sensing and all p-value is less than 0.05. Hence we make a conclusion that the Intuition programmers can write more efficient code than Sensing programmers.

The third category of personality is the “Decision” type. Some people would like to independently think about problems based on basic truth or principles while someone would like to make decisions based on weighing what people care about in a specific situation [4]. Hence, we propose the hypothesis that “Thinking” students are able to write more efficient code than “Feeling” students. This proposal was tested with 7 projects and in Table 8 the running time of code written by Thinking students is significantly shorter than running time of code written by Feeling student because of the corresponding p-value is less than 0.05.

The “Structure” category of the personality describes how people like to live in the outer life. Someone would prefer a planned or orderly way of life while someone like a flexible way of life. Based on the data from 7 projects, we found that Perceiving programmers are more efficient in

writing code than Judging programmers. Since all p-values are less than 0.05, we make a conclusion that statistically the Perceiving programmers outperforms the Judging programmers.

Table 8. Impact of personality subscales on coding performance

Personality	Project	Index	Running time(second)	P-value(one-tail)
Favorite World	1	Extraversion	Mean 0.000816, Std. 0.0048, n=158	0.0298
		Introversion	Mean 0.001627, Std. 0.001576, n=71	
	2	Extraversion	Mean 0.005567, Std. 0.0340, n=159	0.0380
		Introversion	Mean 0.000469, Std. 0.000998, n=74	
	3	Extraversion	Mean 0.01136, Std. 0.0431, n=143	0.0121
		Introversion	Mean 0.00275, Std. 0.009578, n=67	
	4	Extraversion	Mean 0.1724, Std. 0.1754, n=77	0.3826
		Introversion	Mean 0.1813, Std. 0.1420, n=42	
	5	Extraversion	Mean 0.01545, Std. 0.0406, n=132	0.0219
		Introversion	Mean 0.007763, Std. 0.0102, n=56	
	6	Extraversion	Mean 0.4826, Std. 1.7265, n=106	0.0387
		Introversion	Mean 0.1816, Std. 0.1276, n=45	
	7	Extraversion	Mean 0.6963, Std. 0.3659, n=95	0.0319
		Introversion	Mean 0.5802, Std. 0.3263, n=44	
Information	1	Sensing	Mean 0.002, Std. 0.00478, n=174	5.81E-07
		Intuition	Mean 1.73E-04, Std. 2.29E-04, n=55	
	2	Sensing	Mean 0.0065, Std. 0.0343, n=174	0.0085
		Intuition	Mean 2.08E-04, Std. 3.09E-04, n=58	
	3	Sensing	Mean 0.0116, Std. 0.0421, n=158	0.00924
		Intuition	Mean 0.0028, Std. 0.0114, n=52	
	4	Sensing	Mean 0.2056, Std. 0.1645, n=91	1.45E-05
		Intuition	Mean 0.0778, Std. 0.1189, n=28	
	5	Sensing	Mean 0.0225, Std. 0.0612, n=147	1.05E-04
		Intuition	Mean 0.00320, Std. 0.00406, n=41	
	6	Sensing	Mean 0.4927, Std. 1.653, n=118	0.0351
		Intuition	Mean 0.2124, Std. 0.1136, n=33	
	7	Sensing	Mean 0.6996, Std. 0.391, n=104	5.91E-04
		Intuition	Mean 0.4345, Std. 0.3994, n=35	
Decisions	1	Thinking	Mean 0.0017, Std. 0.0045, n=198	0.039
		Feeling	Mean 0.0029, Std. 0.003, n=31	
	2	Thinking	Mean 0.005, Std. 0.032, n=200	0.0141
		Feeling	Mean 0.083, Std. 0.194, n=33	
	3	Thinking	Mean 0.0104, Std. 0.0104, n=180	0.0237
		Feeling	Mean 0.0037, Std. 0.008, n=30	
	4	Thinking	Mean 0.179, Std. 0.179, n=101	0.0157
		Feeling	Mean 0.382, Std. 0.363, n=18	
	5	Thinking	Mean 0.019, Std. 0.0193, n=162	0.0158
		Feeling	Mean 0.1698, Std. 0.3365, n=26	
	6	Thinking	Mean 0.478, Std. 0.478, n=126	0.0158

		Feeling	Mean 0.9078, Std. 0.676, n=25	
	7	Thinking	Mean 0.6919, Std. 0.692, n=115	0.00667
		Feeling	Mean 1.033, Std. 0.608, n=24	
Structure	1	Judging	Mean 0.002, Std. 0.00503, n=154	6.67E-07
		Perceiving	Mean 0.00019, Std. 0.00037, n=75	
	2	Judging	Mean 0.0072, Std. 0.0364, n=154	0.01
		Perceiving	Mean 0.0003, Std. 0.000862, n=79	
	3	Judging	Mean 0.0113, Std. 0.0353, n=136	1.86E-04
		Perceiving	Mean 0.00023, Std. 0.0082, n=74	
	4	Judging	Mean 0.2415, Std. 0.1685, n=74	2.5E-12
		Perceiving	Mean 0.067, Std. 0.2082, n=45	
	5	Judging	Mean 0.0267, Std. 0.066, n=122	5.3E-05
		Perceiving	Mean 0.00264, Std. 0.003, n=66	
	6	Judging	Mean 0.345, Std. 0.358, n=100	0.0338
		Perceiving	Mean 0.2749, Std. 0.092, n=51	
	7	Judging	Mean 0.715, Std. 0.7148, n=87	0.0051
		Perceiving	Mean 0.559, Std. 0.301, n=52	

In some cases, one student writes more efficient codes than another student, it was possible that both students may get the same score or that the grade of the student with more efficient codes may be lower than the grade with less efficient codes. Hence, the second evaluation work was conducted in the impact of personality on academic performance.

For “Favorite World” category, grades of all projects in Introversion are significantly higher than grades in Extraversion. Since Extraversion students like to discuss questions with others, the programming habits and styles are less consistent compared to the coding styles written by Introversion students. Hence, the grades with Extraversion deteriorate when the rubrics were used to evaluate the academic performance

For “Information” category, “Sensing” programmers pay attention to physical reality. So, in our empirical experiment, they focus on the rubrics requirement. But “Intuition” programmers would like to pay more attention to the correctness of code. Hence, as Table 9 showed, Sensing students outperforms Intuition students in projects grades.

For “Decision” category, obviously, we reasonably believe that “Thinking” students can achieve a better grade than “Feeling” students. Data in Table 9 support our hypothesis. With a four-project test, the “Thinking” students got a higher grade than “Feeling” students, which is supported by a p-value of less than 0.05. Hence, we make a conclusion that “Thinking” programmer outperforms “Feeling” programmers in program grading work.

For “Structure” category, “Judging” students prefer a planned or orderly way of life such as having things settled or organized. Specifically in project tests, they feel comfortable if they plan their code style organized. But for “Perceiving” students, they prefer a flexible life. When they plan to finish the projects, the correctness is the priority instead of the requirement of rubrics. As Table 9 showed, the p-value in first 3 projects was less than 0.05. In project 4 although “Perceiving” students achieved a better grade than “Judging” students, the p-value is greater than 0.05. So statistically the grade in Perceiving was not significantly better than the grade in Judging. Based on the analysis above, we make a conclusion that “Judging” programmers can achieve a better grade than “Perceiving” programmers.

Table 9. Impact of personality subscales on academic performance

Personality	Assignment	Index	Grade	P-value(one-tail)
Favorite World	1	Extraversion	Mean 88.34, Std. 15.13, n=182	0.0193
		Introversion	Mean 91.70, Std. 10.53, n=83	
	2	Extraversion	Mean 79.11, Std. 22.35, n=184	0.017

		Introversion	Mean 84.18, Std. 15.50, n=82	
	3	Extraversion	Mean 83.09, Std. 22.51, n=171	0.037
		Introversion	Mean 87.77, Std. 16.81, n=74	
	4	Extraversion	Mean 83.27, Std. 20.15, n=172	0.0025
		Introversion	Mean 89.85, Std. 14.39, n=69	
Information	1	Sensing	Mean 88.95, Std. 14.34, n=197	0.0037
		Intuition	Mean 80.99, Std. 22.43, n=68	
	2	Sensing	Mean 80.58, Std. 20.32, n=195	0.0104
		Intuition	Mean 73.04, Std. 23.99, n=70	
	3	Sensing	Mean 83.58, Std. 21.78, n=186	0.0456
		Intuition	Mean 77.16, Std. 26.10, n=59	
	4	Sensing	Mean 85.28, Std. 18.75, n=179	0.0258
		Intuition	Mean 78.49, Std. 24.73, n=62	
Decisions	1	Thinking	Mean 89.07, Std. 14.94, n=228	0.2273
		Feeling	Mean 87.35, Std. 12.51, n=37	
	2	Thinking	Mean 79.41, Std. 21.11, n=228	0.0124
		Feeling	Mean 70.14, Std. 22.80, n=37	
	3	Thinking	Mean 83.53, Std. 22.23, n=212	0.0038
		Feeling	Mean 69.90, Std. 26.90, n=34	
	4	Thinking	Mean 84.06, Std. 20.10, n=206	0.0387
		Feeling	Mean 76.07, Std. 24.75, n=35	
Structure	1	Judging	Mean 89.85, Std. 13.62, n=173	0.0014
		Perceiving	Mean 82.48, Std. 21.00, n=92	
	2	Judging	Mean 79.86, Std. 21.08, n=174	0.0338
		Perceiving	Mean 74.49, Std. 23.42, n=92	
	3	Judging	Mean 83.73, Std.21.84, n=161	0.0140
		Perceiving	Mean 76.29, Std. 26.49, n=85	
	4	Judging	Mean 83.96, Std. 20.70, n=159	0.2840
		Perceiving	Mean 85.40, Std. 17.41, n=82	

We also explored whether there exists a linear correlation between each category of the personality and the coding performance in Table 10. Since we converted the value of Introversion to the value of Extraversion in Personality row (Table 10), we use EI' to denote the conversion from Introversion to Extraversion. The similar conversion can be applied in the rest of three categories: SN', TF' and JP'.

The rules in the correlation are explained as follows: taking EI' column as an example, if the correlation value is a positive number, the coding performance is linearly relative to

Extraversion. If the correlation value is a negative number, the coding performance is linearly relative to the opposite category of Extraversion: Introversion. Also, this explanation of correlation numbers was applied to the rest of three categories: SN', TF' and JP'.

In " EI' " column of Table 10, the coding performance in most projects is, negatively, linearly dependent to the Extroversion character. Namely, the coding performance increases when the weight of Introversion character gets higher. And the coding performance becomes deteriorated when programmers are more and more extraverted. But an exception happened to the project 7. The correlation value is 0.6801, which is less than 0.7. In Matlab course, there are 7 projects assigned to students and the last one is complicated, time-consuming assignment. So, it suppresses the distinguish in coding performance.

In " SN' " column, the correlation values from all projects show that the coding performance is, positively, linearly dependent to Sensing instead of Intuition. And in " TF' " column, we found the coding performance rises when programmers prefer to solve problems in a "Thinking" way instead of a "Feeling" way. Although the experimental value of 0.6955 is made in project 2, we still believe our conclusion that the coding performance is, positively, linearly dependent to the "Thinking" category.

In " JP' " column, the students with a planned or orderly way of life obviously outperforms ones who prefer a flexible way of life in coding performance. All values in " JP' " column is greater than 0.7 so we reasonably believe that the coding performance is, positively, linearly dependent to Judging characteristic.

Table 10. Correlation of influence of personality subscales on coding performance

Personality / Project	EI'	SN'	TF'	JP'
1	-0.7986	0.8111	0.7137	0.7663
2	-0.7121	0.8350	0.6955	0.7332
3	-0.7467	0.9425	0.7831	0.7690
4	-0.7172	0.9429	0.7290	0.7523
5	-0.7990	0.9534	0.8129	0.8346
6	-0.7701	0.8346	0.7173	0.7729
7	-0.6801	0.8740	0.7380	0.7358

Additionally, we also explored the correlation between students' academic performance and the personality. For "Favorite World" category, we found there exists a strong linear dependence between students' grade and their Introversion character in project 2 and project 3. Unfortunately, we did not find this linear dependence in project 1 and project 4.

For "Information" category, since students with Sensing character would like to pay attention to the physical reality, their grade is highly linearly dependent to Sensing if the rubrics were presented by the instructor. Also, for "Structure" and "Decision" categories, the academic performance is positively linearly dependent to "Thinking" and "Judging".

Table 11. Correlation of influence of personality subscales on academic performance

Personality / Assignment	EI'	SN'	TF'	JP'
1	-0.05496	0.933958	0.847646	0.789044
2	-0.75081	0.949684	0.831769	0.831429
3	-0.72657	0.945148	0.790098	0.778046
4	-0.12858	0.894694	0.744781	0.906151

To determine the strength of the relationship between the coding performance or the academic performance and a series of variables in the personality, we also utilized the multiple linear regressions to predict the impact of each category of personality on the coding performance and academic performance in Table 12 and Table 13. So that it can help educators and researchers to value: (1) personality categories, and (2) the relationship among personality subscales and their coding performance or academic performance.

In Table 12, totally 58 students correctly completed all projects and the R-square is 0.96. Although R-square can describe how close the data are to the fitted regression line, the residual plots need to be checked before we make a conclusion about our prediction model in (1) coding performance and personality; (2) academic performance and personality. Residual plots can reveal unwanted residual patterns that indicate biased results more effectively than numbers. Here we used statistical terms to define residual: the difference between the observed value of the dependent variable and the predicted value.

In Fig. 5, the independent variable Extraversion is denoted on the horizontal axis and the residual is showed on the vertical axis. Clearly, all points are randomly dispersed around the horizontal axis. Hence, the linear model of Extraversion is appropriate for that data. Similarly, in Figs. 6-8, most points are randomly distributed around the horizontal axis. Based on this observation, we make a conclusion that the linear regression model in Table 12 is valid for personality and coding performance.

Table 12. Regression of influence of personality subscale on coding performance

Project		Coefficients	P-value	R Square	N
Performance	Intercept	0.38	2.42E-04	0.96	58
	EI'	-0.007	2.47E-07		
	SN'	8.4E-04	0.18		
	TF'	0.0032	0.0047		
	JP'	-7.4E-04	0.45		
Regression Equation	-0.007*E+8.4E-04*S+0.032T-7.4E-04*J+0.38				

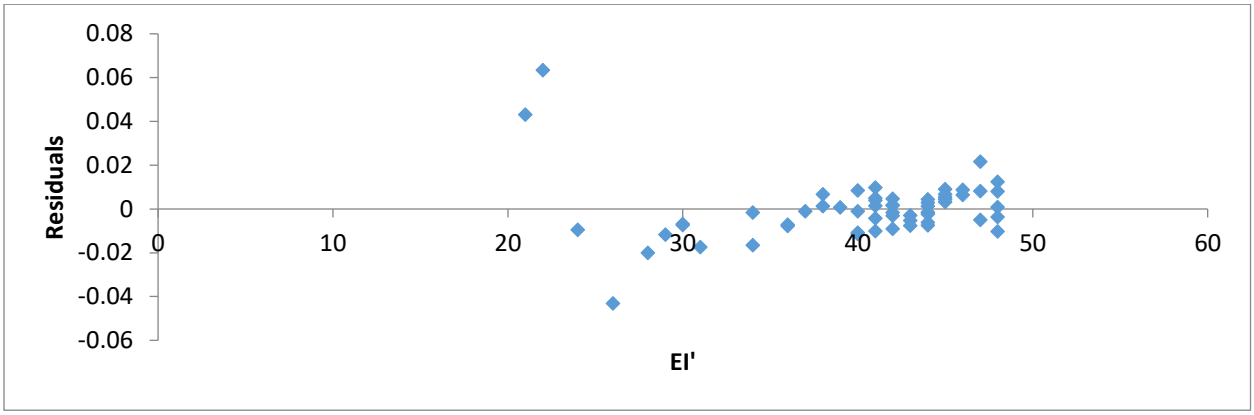

Figure 5. Extraversion vs. Introversion residual plot for coding performance

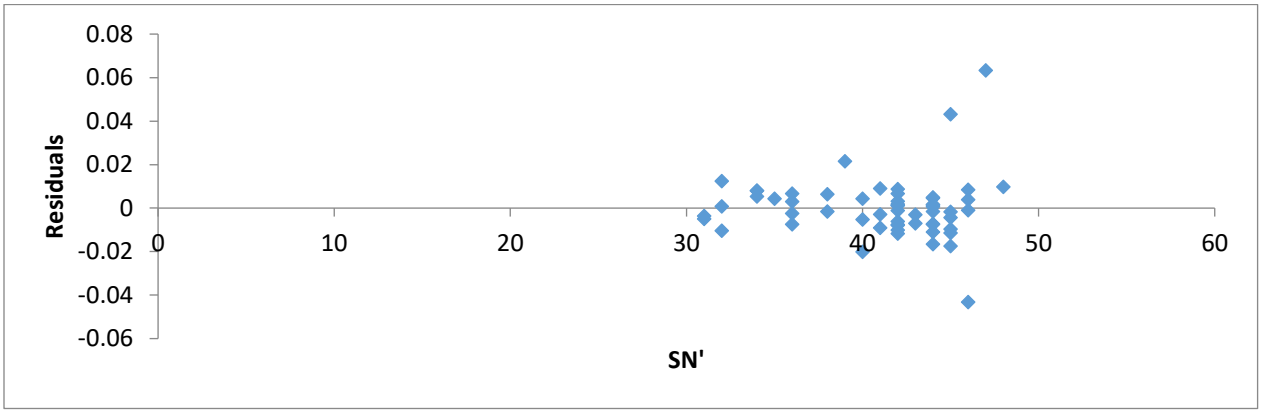

Figure 6. Sensing vs. Intuition residual plot for coding performance

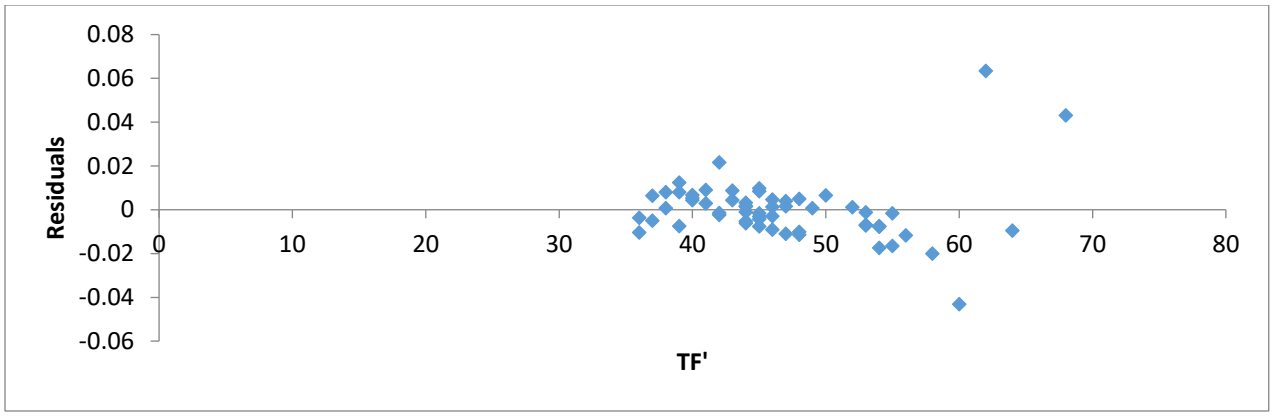

Figure 7. Thinking vs. Feeling residual plot for coding performance

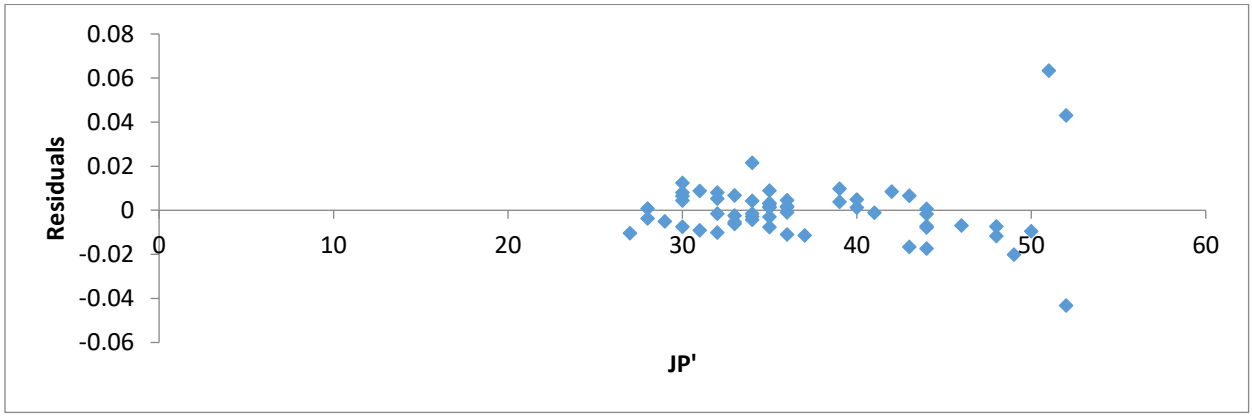

Figure 8. Judging vs. Perceiving residual plot for coding performance

The data in Table 13 present a prediction model between personality and academic performance and the R-square is 0.87. In Figs. 9-12, although some points are below or above the horizontal axis, overall all points are still randomly dispersed around x-axis. We concluded that our prediction model was valid to predict the academic performance based on the personality.

Table 13. Regression of influence of personality subscale on academic performance

Project		Coefficients	P-value	R Square	N
Grade	Intercept	0.51	8.37E-41	0.87	208
	EI'	0.0048	4.58E-05		
	SN'	0.0024	0.068		

	TF'	-0.0058	3.47E-04		
	JP'	0.0065	1.82E-06		
Regression Equation	0.0048*E+0.0024*S-0.0058*T+0.0065*J				

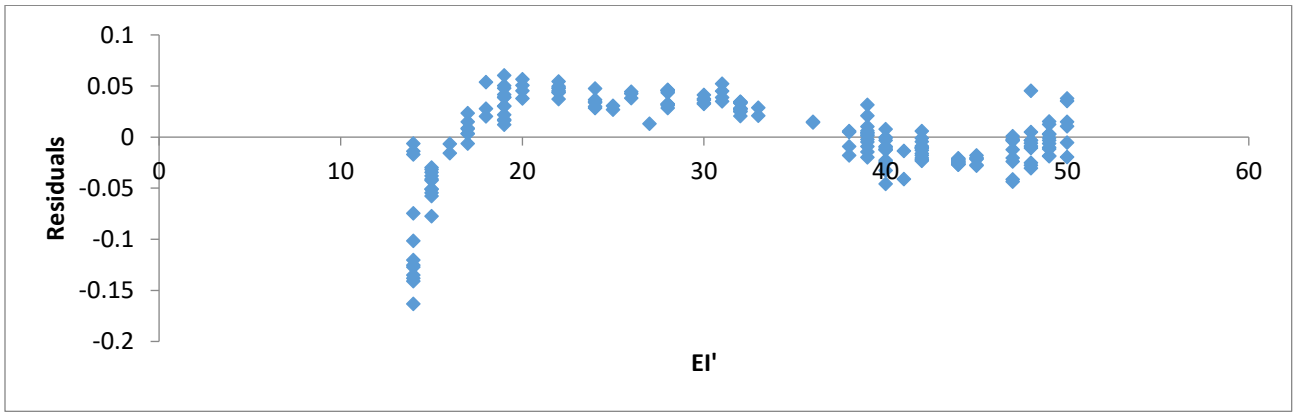

Figure 9. Extraversion vs. Introversion residual plot for academic performance

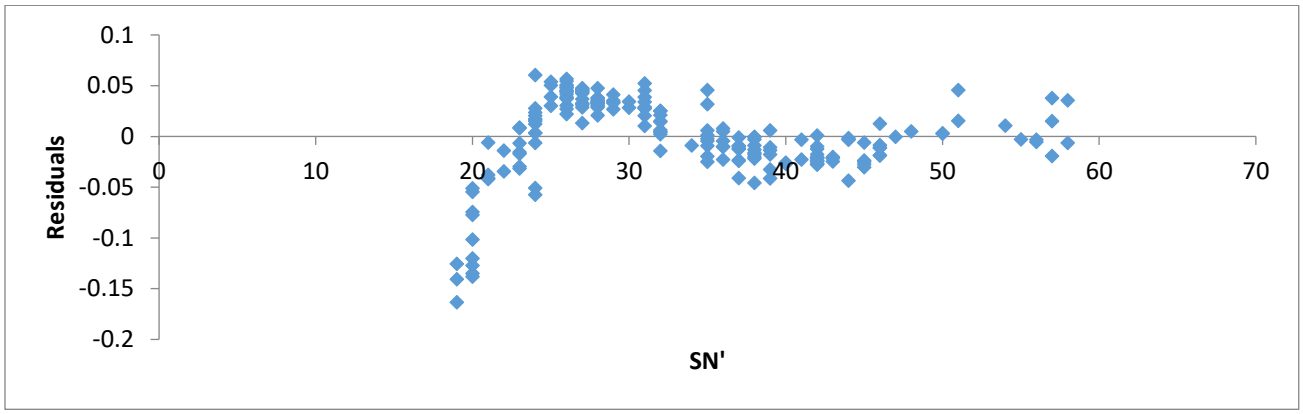

Figure 10. Sensing vs. Intuition residual plot for academic performance

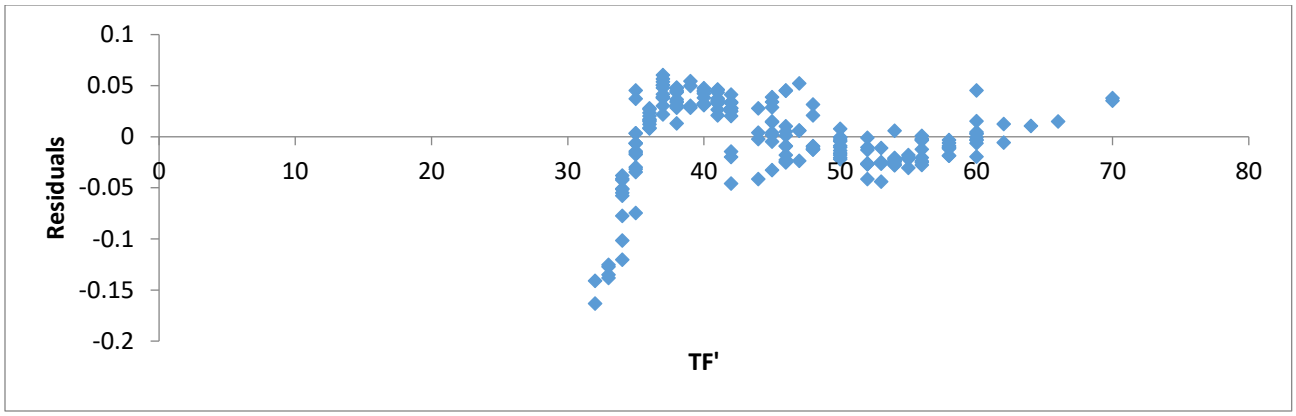

Figure 11. Thinking vs. Feeling residual plot for academic performance

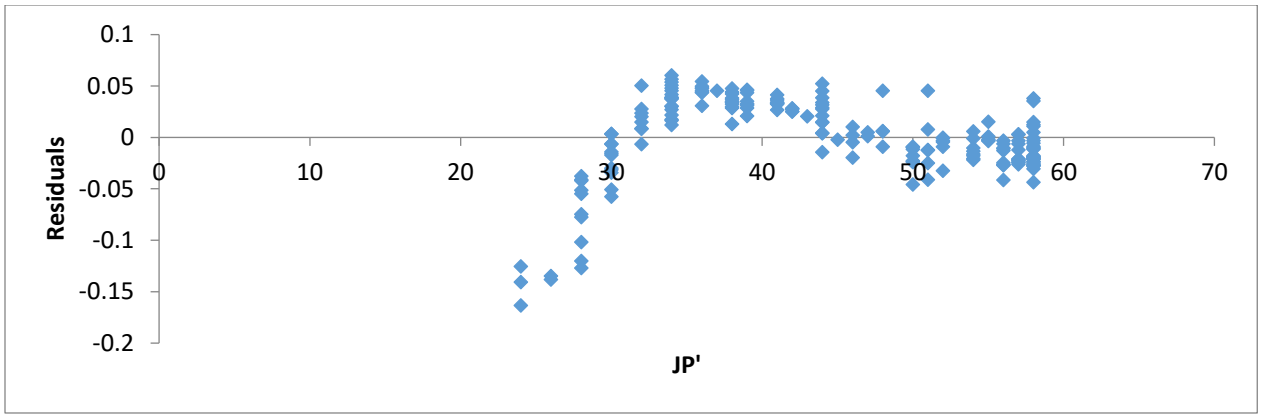

Figure 12. Judging vs. Perceiving residual plot for academic performance

4.5.2 Influence of programming attitude on performance

Since there are five subscales of programming attitude, we respectively analyzed the impact of each subscale on the performance. Also under each subscale, the performance was compared with p-value of independent sample T-test to explore whether the significant difference in the performance exists or not. Before we explain data in this section, the acronym

in the tables and figures needs to be clarified: (1) CNC'—Confidence and Non-confidence; (2) ATNAT' – Attitude toward success and Attitude toward nonsuccess; (3) MDFD' – Male domain and Female domain; (4) UFUL' – Useful and Non-useful; (5) EMIM' – Effective Motion and Ineffective Motion. In Table 14 the data show the coding performance based on the running time under each subscale with 7 projects.

For "Confidence in learning computer science and programming" category, the participants with confidence characteristic can write more efficient code than participants with non-confidence characteristic because the running time with Confidence character is less than the running time with Non-confidence character in all projects (Table 14) and the corresponding p-value is less than 0.05. Hence statistically we made a conclusion that the Confidence programmers outperform the Non-confidence programmers.

For "Attitude toward success in computer science" category, we proposed the programmers with success attitude would write more efficient code than programmers who hesitate to finish the projects. Table 14 shows that the running time from "Success attitude" students is significantly shorter than one from "Non-success attitude" students in 7 projects with a less than 0.05 p-value. Hence, our hypothesis is supported by the p-value from 7 projects.

For "Computer science as a male domain" category, students who believe the computer science is dominated by male domain outperform students who believe the computer science is dominated by female domain based on the running time index. The data in project 6 of Table 14

failed to be collected because all participants agreed with a "male domain" selection. The rest of projects supported our hypotheses by p-value which is less than 0.05.

For both "Usefulness of computer science and programming" and "Effective motivation in computer science and programming" categories, we propose students believing computer science is useful and effective in the life can write more efficient code than ones believing computer science is non-useful and ineffective in the life. Although in project 6 the code from effective students is not significantly more efficient than code from ineffective student, the rest of 6 projects still has a less-than-0.05 p-value. Hence, the experimental results support our hypothesis.

Table 14. Impact of attitude subscale on coding performance

Attitude	Project	Index	Running time(second)	P-value(one-tail)
Confidence in learning computer science and programming	1	Confidence	Mean 1.88E-04, Std. 3.39E-04, n=118	1.97E-04
		Unconfident	Mean 0.0011, Std. 0.0029, n=122	
	2	Confidence	Mean 4.17E-04, Std. 9.83E-04, n=122	5.34E-10
		Unconfident	Mean 0.0021, Std. 0.0026, n=118	
	3	Confidence	Mean 8.91E-04, Std. 0.0030, n=112	0.013
		Unconfident	Mean 0.011, Std. 0.047, n=112	
	4	Confidence	Mean 0.0528, Std. 0.073, n=85	9.01E-07
		Unconfident	Mean 0.1977, Std. 0.2144, n=64	
	5	Confidence	Mean 0.0028, Std. 0.0036, n=101	0.031
		Unconfident	Mean 0.0073, Std. 0.0209, n=82	
	6	Confidence	Mean 0.2239, Std. 0.0980, n=81	5.31E-04
		Unconfident	Mean 0.2768, Std. 0.0933, n=66	
	7	Confidence	Mean 0.7378, Std. 0.0847, n=76	0.0183
		Unconfident	Mean 1.044, Std. 1.127, n=62	
Attitude toward success in computer science	1	Success	Mean 6.72E-04, Std. 0.0021, n=235	0.004
		Unsuccess	Mean 0.0010, Std. 5.56E-05, n=5	
	2	Success	Mean 0.0012, Std. 0.00213, n=235	0.0146
		Unsuccess	Mean 0.0035, Std. 0.0015, n=2	
	3	Success	Mean 0.0060, Std. 0.0341, n=219	0.0204
		Unsuccess	Mean 0.055, Std. 0.0363, n=5	
	5	Success	Mean 0.0048, Std. 0.0145, n=180	0.0379
		Unsuccess	Mean 0.1042, Std. 0.0503, n=3	
	6	Success	Mean 0.4000, Std. 1.4804, n=143	0.0075

		Unsuccess	Mean 1.9610, Std. 0.7234, n=4	
	7	Success	Mean 0.6801, Std. 0.3800, n=135	0.0423
		Unsuccess	Mean 2.1350, Std. 0.7819, n=3	
Computer science as a male domain	1	Male domain	Mean 6.49E-04, Std. 0.0021, n=237	0.0078
		Female domain	Mean 0.0074, Std. 0.0015, n=3	
	2	Male domain	Mean 0.0012, Std. 0.0021, n=237	0.0148
		Female domain	Mean 0.0078, Std. 0.002, n=3	
	3	Male domain	Mean 0.0060, Std. 0.033, n=221	0.0087
		Female domain	Mean 1.273, Std. 0.531, n=4	
	4	Male domain	Mean 0.118, Std. 0.1700, n=147	0.0246
		Female domain	Mean 0.308, Std. 0.072, n=3	
	5	Male domain	Mean 0.0049, Std. 0.015, n=180	0.0318
		Female domain	Mean 0.055, Std. 0.023, n=3	
	7	Male domain	Mean 0.6810, Std. 0.3787, n=136	0.0246
		Female domain	Mean 1.86, Std. 0.121, n=2	
Usefulness of computer science and programming	1	Useful	Mean 2.07E-04, Std. 4.83E-04, n=182	1.9E-04
		Unuseful	Mean 0.0021, Std. 0.0039, n=58	
	2	Useful	Mean 3.99E-04, Std. 0.001, n=181	1.59E-15
		Unuseful	Mean 0.0038, Std. 0.0025, n=59	
	3	Useful	Mean 0.0053, Std. 0.038, n=169	0.044
		Unuseful	Mean 0.097, Std. 0.3935, n=56	
	4	Useful	Mean 0.065, Std. 0.082, n=117	7.85E-06
		Unuseful	Mean 0.290, Std. 0.253, n=33	
	5	Useful	Mean 0.0027, Std. 0.0034, n=145	0.022
		Unuseful	Mean 0.0129, Std. 0.030, n=38	
	6	Useful	Mean 0.426, Std. 1.651, n=115	0.018
		Unuseful	Mean 0.93, Std. 1.0084, n=32	
	7	Useful	Mean 0.642, Std. 0.241, n=116	0.016
		Unuseful	Mean 1.03, Std. 0.792, n=22	
Effective motivation in computer science and programming	1	Effective	Mean 2.77E-04, Std. 0.0014, n=130	0.0013
		Ineffective	Mean 0.0011, Std. 0.0025, n=110	
	2	Effective	Mean 4E-04, Std. 9.56E-04, n=134	7.77E-11
		Ineffective	Mean 0.0023, Std. 0.0026, n=106	
	3	Effective	Mean 8.91E-04, Std. 0.0031, n=129	0.013
		Ineffective	Mean 0.0126, Std.0.0508, n=96	
	4	Effective	Mean 0.048, Std. 0.075, n=88	2.97E-08
		Ineffective	Mean 0.216, Std.0.2108, n=62	
	5	Effective	Mean 0.0028, Std. 0.0035, n=113	0.025
		Ineffective	Mean 0.0082, Std. 0.0226, n=70	
	6	Effective	Mean 0.479, Std. 1.875, n=89	0.147
		Ineffective	Mean 0.269, Std. 0.0960, n=58	
	7	Effective	Mean 0.631, Std. 0.2337, n=82	0.01
		Ineffective	Mean 0.857, Std. 0.6823, n=56	

Similarly, we also explore the relationship between the academic performance and the personality. In the following categories: (1) Confidence in learning computer science and programming; (2) Attitude toward success in computer science; (3) Usefulness of computer science and programming; and (4) Effective motivation in computer science and programming, the results with p-value in Table 15 show that students with positive attitude in each category can significantly finish projects with higher grade than students with negative attitude. But in the "Computer science as a male domain" attitude, all students agreed with the option of male-dominate-computer science so the experiment failed to be conducted. Especially in project 2, since p-value (0.155) is greater than 0.05, we are not able to make a conclusion that male programmers can attitude can achieve a better grade than female programmer.

Table 15. Impact of attitude subscale on academic performance

Attitude	Assignment	Index	Grade	P-value(one-tail)
Confidence in learning computer science and programming	1	Confidence	Mean 88.99, Std. 16.13, n=136	0.018
		Unconfident	Mean 84.89, Std. 84.89, n=135	
	2	Confidence	Mean 82.95, Std. 18.93, n=135	0.0043
		Unconfident	Mean 76.07, Std. 23.35, n=133	
	3	Confidence	Mean 88.54, Std. 16.10, n=128	5.1E-05
		Unconfident	Mean 77.69, Std. 25.26, n=117	
	4	Confidence	Mean 87.33, Std. 17.13, n=127	0.0032
		Unconfident	Mean 80.23, Std. 22.56, n=117	
Attitude toward success in computer science	1	Success	Mean 87.90, Std. 15.45, n=265	0.045
		Unsuccess	Mean 68.17, Std. 22.99, n=6	
	2	Success	Mean 80.10, Std. 20.45, n=261	0.0478
		Unsuccess	Mean 54.67, Std. 30.21, n=6	
	3	Success	Mean 83.63, Std. 21.37, n=239	0.0076
		Unsuccess	Mean 57.33, Std. 17.43, n=6	
	4	Success	Mean 84.44, Std. 19.38, n=238	0.0176
		Unsuccess	Mean 57, Std. 23.26, n=6	
Computer science as a male domain	1	Male domain	Mean 88.42, Std. 15.26, n=267	0.01
		Female domain	Mean 68.25, Std. 15.47, n=4	
	2	Male domain	Mean 79.77, Std. 21.14, n=262	0.155
		Female domain	Mean 82.8, Std. 21.91, n=5	
	4	Male domain	Mean 84.48, Std. 19.51, n=238	0.023
		Female domain	Mean 66.3, Std. 14.02, n=5	

Usefulness of computer science and programming	1	Useful	Mean 88.62, Std. 14.58, n=206	0.0056
		Unuseful	Mean 82.15, Std. 16.76, n=65	
	2	Useful	Mean 81.84, Std. 19.37, n=200	0.008
		Unuseful	Mean 73.85, Std. 22.80, n=67	
	3	Useful	Mean 85.68, Std. 20.33, n=185	0.0035
		Unuseful	Mean 76.19, Std. 24.01, n=60	
	4	Useful	Mean 86.64, Std. 17.53, n=187	0.0017
		Unuseful	Mean 76.38, Std. 23.48, n=56	
Effective motivation in computer science and programming	1	Effective	Mean 88.05, Std. 17.16, n=153	0.022
		Ineffective	Mean 84.25, Std. 13.47, n=118	
	2	Effective	Mean 82.54, Std. 18.31, n=149	0.011
		Ineffective	Mean 76.42, Std. 23.43, n=118	
	3	Effective	Mean 85.42, Std. 20.15, n=142	0.044
		Ineffective	Mean 80.52, Std. 23.31, n=103	
	4	Effective	Mean 86.43, Std. 18.05, n=139	0.025
		Ineffective	Mean 81.39, Std. 21.03, n=104	

In Table 16, the results answered the research problem that whether there exists a strong linear dependence between attitude and coding performance. In the “Confidence” category, all p-value from 7 projects is greater than 0.7, which supports our hypothesis: the confidence attitude is positively linearly dependent to the coding performance. Similarly, in “Effective”, “Success” and “Usefulness” categories, although some p-value is slightly less than 0.7, we still make a conclusion that there exists a strong linear dependence between the positive attitude in each category and the coding performance. Unfortunately, in the “computer science as a male domain” category we did not find any strong linear dependence between the coding performance and participants’ gender.

Table 16. Correlation of influence of attitude subscale on coding performance

Attitude / Project	**ATNAT'**	**MDFD'**	**UFUL'**	**EMIM'**	**CNC'**
1	0.8226	0.1371	0.8833	0.8978	0.9656
2	0.8360	0.0804	0.8632	0.7999	0.8514
3	0.6252	0.1117	0.6871	0.7047	0.8083
4	0.8816	0.0753	0.9358	0.7468	0.9460
5	0.9008	0.0678	0.9052	0.8048	0.9620
6	0.91290	0.1024	0.9383	0.7804	0.9590
7	0.7318	0.1459	0.8067	0.6650	0.8151

In Table 17, the hypothesis that there is a strong linear dependence between each subscale of attitude and academic performance is also explored. Under five subscales of attitude, we can find that in "computer science as a male domain" category, all p-value in the 4 assignments is less than 0.7. Hence, we make a conclusion that the academic performance is not linearly dependent to participants' gender. But for the rest of four subscales, all analysis from 4 assignments supports our original hypothesis. Hence, we can make a conclusion based on data from Table 17: the positive dimension in each category of attitude except the "computer science as a male domain" category is strongly linearly dependent to the academic performance.

Table 17. Correlation of influence of attitude subscale on academic performance

Attitude / Assignment	ATNAT'	MDFD'	UFUL'	EMIM'	CNC'
1	0.7770	0.1225	0.8028	0.8006	0.8309
2	0.8338	0.0218	0.8394	0.8139	0.8589
3	0.8250	0.0735	0.8221	0.7414	0.8030
4	0.8219	0.1486	0.8802	0.8549	0.9229

Since a linear dependence exploration has been conducted, the prediction model is supposed to be presented, if possible. Firstly, we tried to describe the prediction model between attitude and coding performance. In Table 18 we can see that R-square is 0.67 which is less than 0.7. But before any conclusion can be made, we still needed to check the residual plot for each subscale. In Figures 13 – 17, we can see that all points are randomly dispersed around x-axis so that the regression model in Table 18 fits our data.

Table 18. Regression of influence of attitude subscale on coding performance

Project		Coefficients	P-value	R Square	N
	Intercept	0.076	0.37		
	CNC'	0.0045	0.041		

Performance	ATNAT'	-3.4E-04	0.92	0.67	67
	MDFD'	2.52E-04	0.93		
	UFUL'	0.0013	0.04		
	EMIM'	-0.0025	2.02E-04		
Regression Equation	0.0045*C-3.4E-04*AT+2.52E-04*MD+0.0013*UF-0.0025*EM+0.076				

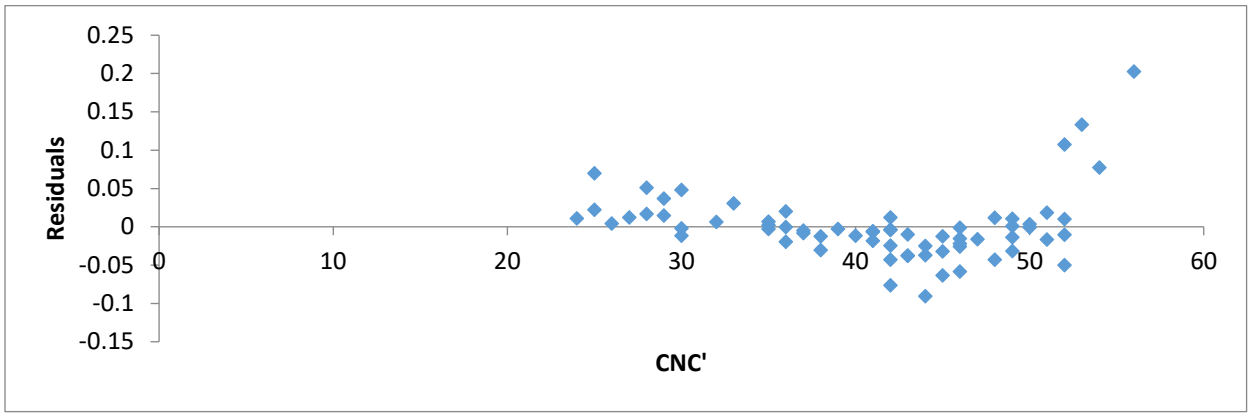

Figure 13. Confidence vs. Non-confidence residual plot for coding performance

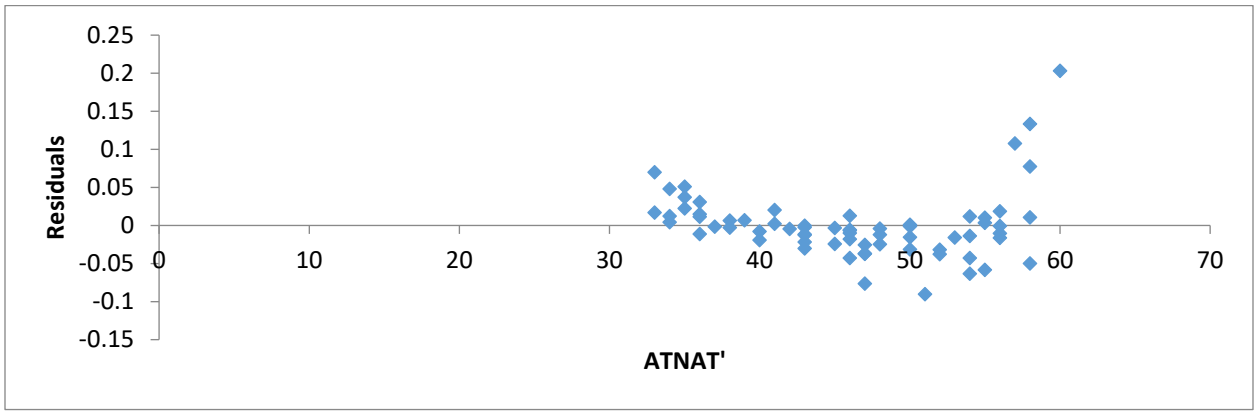

Figure 14. Success vs. Nonsuccess residual plot for coding performance

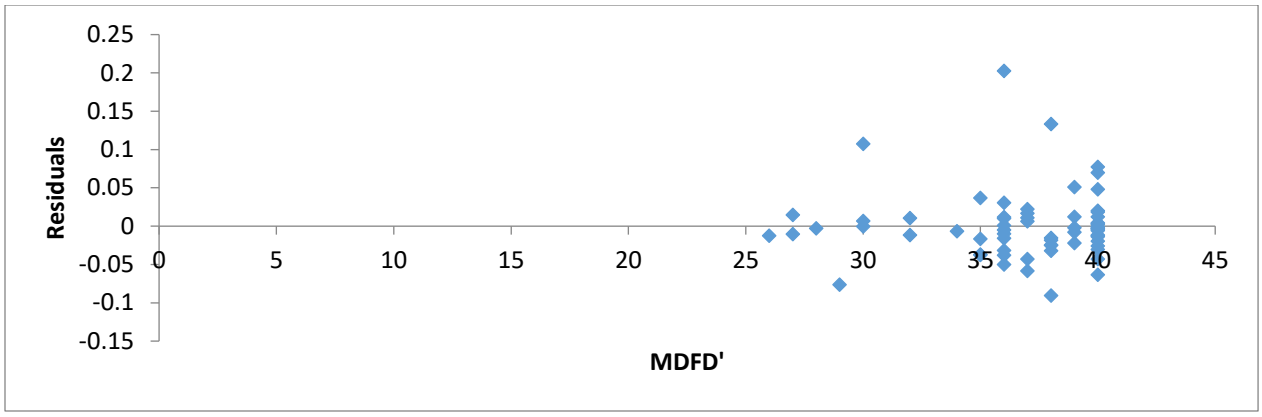

Figure 15. Male-domain vs. Female-domain residual plot for coding performance

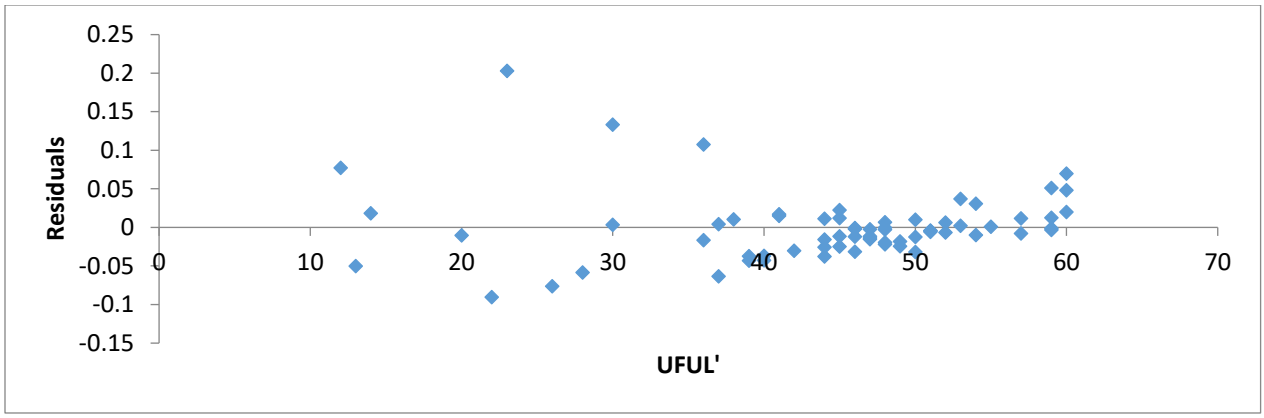

Figure 16. Usefulness vs. Unusefulness residual plot for coding performance

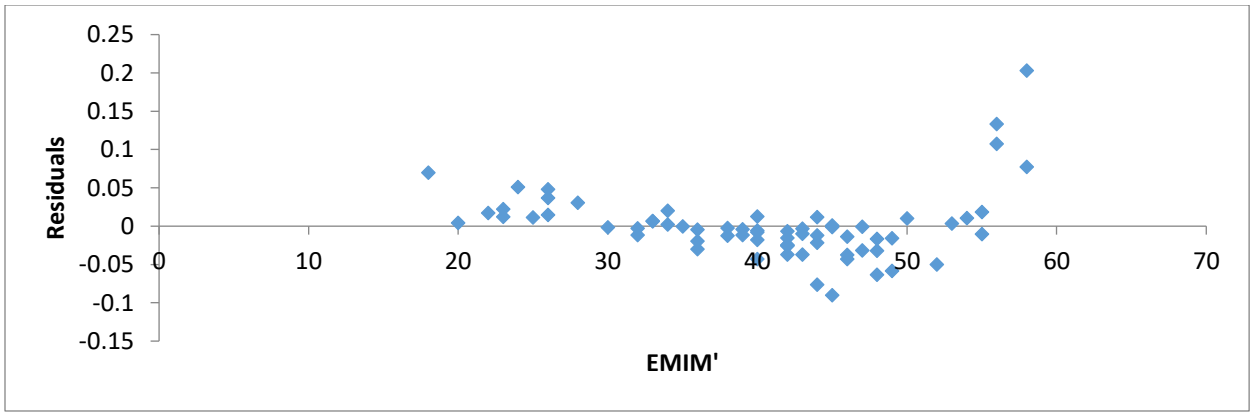

Figure 17. Effectiveness vs. Ineffectiveness residual plot for coding performance

Also, the prediction model was explored between the programming attitude and the academic performance. The R-square in Table 19 is 0.98 which is greater than the statistical standard value of 0.7. But before a conclusion is made, the residual plot of each category also needs to be checked. In Figure 19, few points are located below the horizontal axis but it does not affect our conclusion that our regression model perfectly fits our experimental data.

Table 19. Regression of influence of attitude subscale on academic performance

Project		Coefficients	P-value	R Square	N
Grade	Intercept	0.24	9.15E-51	0.98	206
	CNC'	0.0039	1.38E-15		
	ATNAT'	-0.0015	1.85E-09		
	MDFD'	-1.2E-05	0.95		
	UFUL'	-6.1E-04	0.0066		
	EMIM'	0.009	1.22E-26		
Regression Equation	0.0039*C-0.0015*AT-1.2E-05*MD-6.1E-04*UF+0.009*EM+0.24				

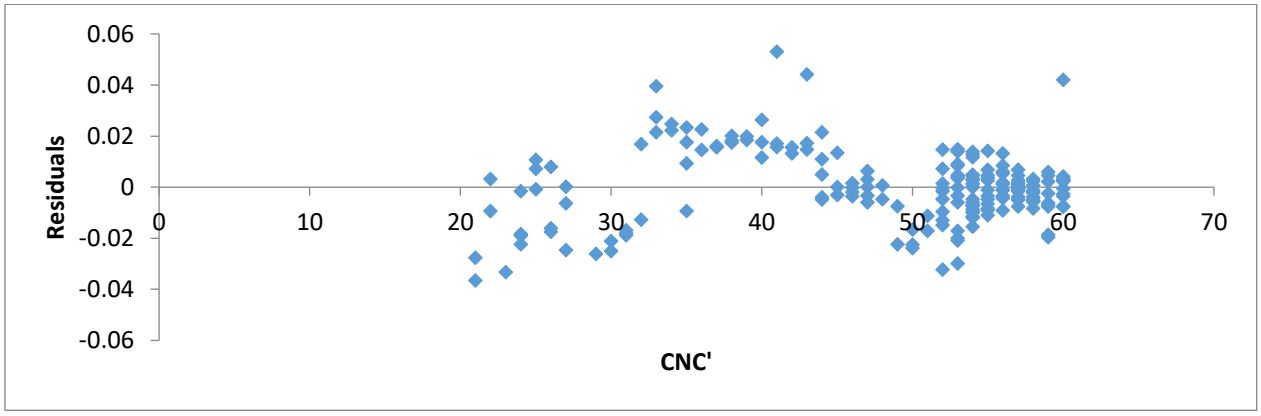

Figure 18. Confidence vs. Non-confidence residual plot for academic performance

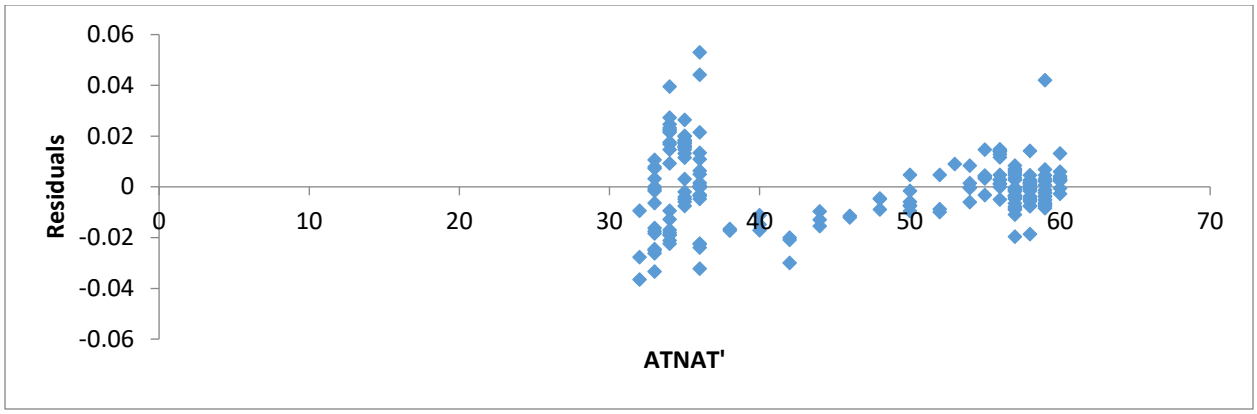

Figure 19. Success vs. Nonsuccess residual plot for academic performance

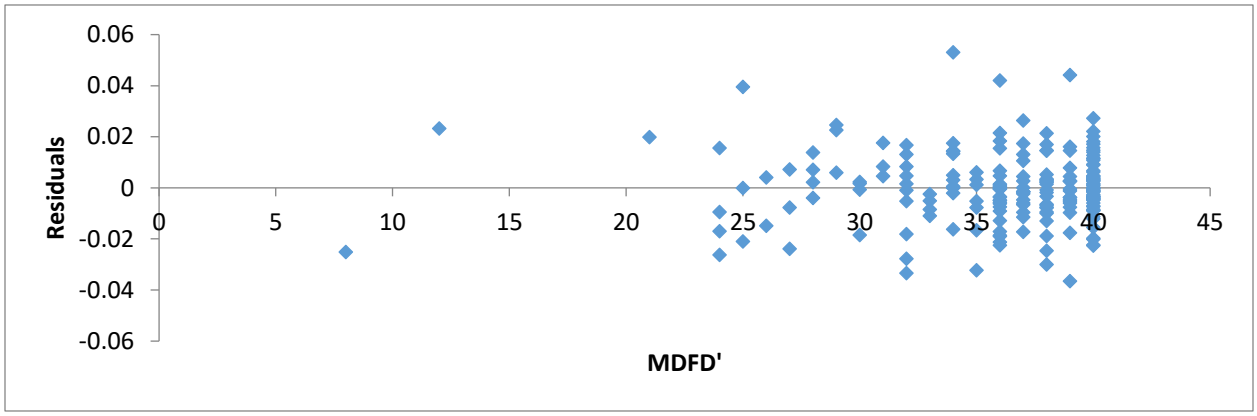

Figure 20. Male-domain vs. Female-domain residual plot for academic performance

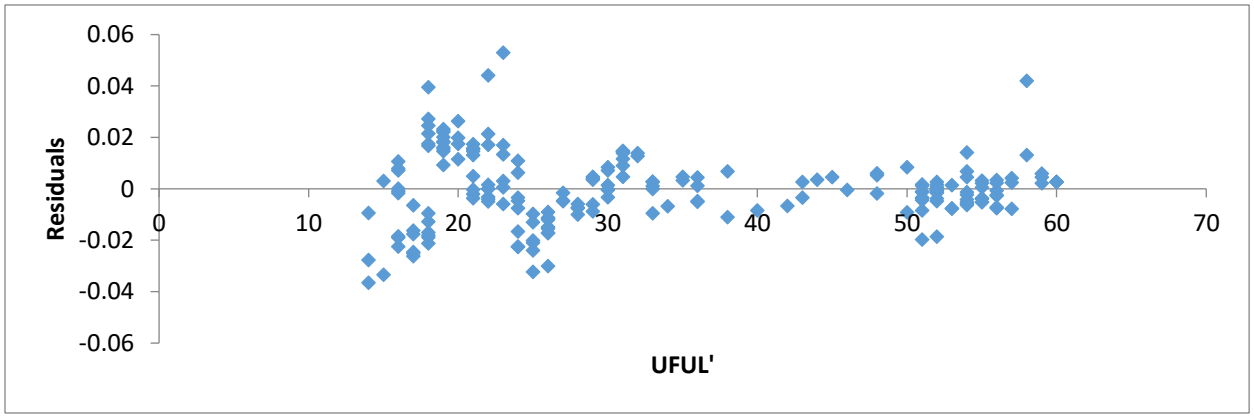

Figure 21. Usefulness vs. Unusefulness residual plot for academic performance

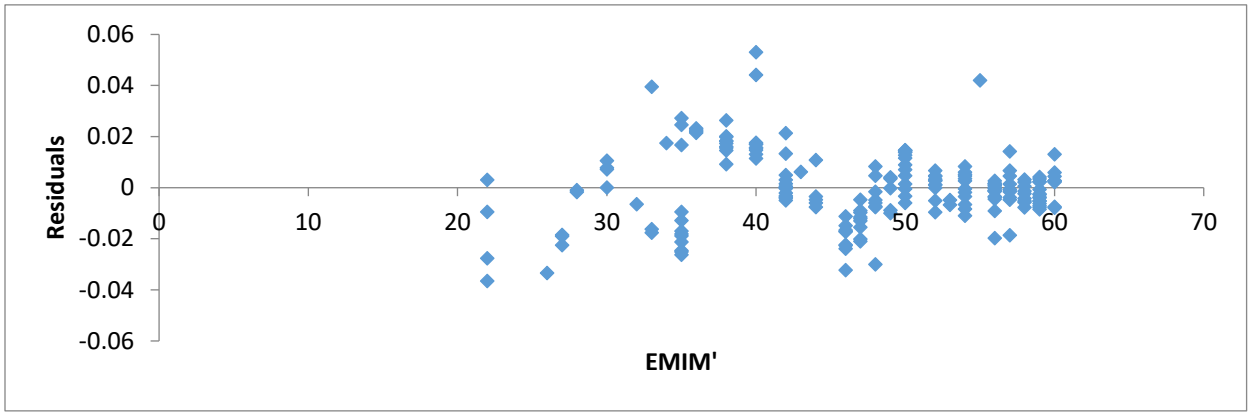

Figure 22. Effectiveness vs. Ineffectiveness residual plot for academic performance

4.5.3 Influence of programming styles on the performance

Since current programming styles were collected since 1985, we developed an updated programming styles after 2006. Based on the literature review work on research papers, experts phone interviews and the topics search online, totally 9 different styles (the first column in Table 20) were created and the survey work had been distributed among 328 students to verify whether the survey development work has acceptable internal consistency or not. In our research, we focus on a statistical problem: how closely related a set of items are as a group. Normally Cronbach's alpha can be an effective measure of the internal consistency.

The results of Cronbach's alpha are showed in Table 20. Among 9 programming styles, we can see that 3 of them cannot be verified because their alpha value is less than 0.7. Statistically, it

cannot be accepted in a research work. But the rest of 6 styles can be verified since all alpha value is greater than 0.7. Hence in the following work we only adopt the 6 verified styles to test our hypotheses instead of original 9 styles.

Table 20. Internal consistency of programming styles

Subscale	Survey Statement Numbers	Cronbach's Alpha	Number of Participants
Alone	7-13	0.73	324
Continuous	14-15	0.81	
Open Source	16-17	0.93	
Visual	18-19	0.76	
Units	20-21	0.96	
Office	22a-22b	0.11	
whiteboard	22c-22d	0.1	
Efficient	22e-22f	0.83	
automatic	22g-22h	0.55	

To answer the research question whether there is a significant performance difference between the positive dimension and the negative dimension of each category or not, we employed an independent T-test sample to analyze it. The data are showed in Table 21 and the conclusions will be accepted with a less than 0.05 p-value.

In "Programming Alone" category, the running time with the "Group" program style is significantly less than the running time with the "Programming Alone" style in all tests. But since the 7th project was complicated, there is no significant difference statistically because the p-value is greater than 0.05 in Table 21. The rest of 6 projects still support our hypothesis: the code written by the group is faster than code written by a single student.

In “Continuous Programming” category, since intermittent programming style would distract programmers’ attention, the finished code may be less efficient than code written by programmers with the “continuous” programming style. Unfortunately, the test data from 7 project did not support our hypothesis. As Table 21 showed, only the data in project 2 and 4 support our original hypothesis and the rest of 5 projects did not show there is significant difference between two different programming style. Hence, there is no significant difference in code efficiency.

In “Open Source maintenance” category, programmers with the “Open Source” styles would like to work with others, so the code may be more efficient than code written by those with a “Closed Source” programming habit. The p-value in project 5 and 6 of Table 21 is greater than 0.05, which means that although the running time of code written by programmers with “Closed Source” style is shorter than the running time of code with a “Open Source” style, it is not a significant difference statistically. Hence a conclusion is made that the “Open Source” programmers can write more efficient code than “Closed Source” programmers.

In the analysis of the visual-based programming style, except the project 6 we can see the finished code under the “visual” programming context is significantly faster than code finished under the “text” programming context. Similarly, in “unit-test” category, although the code with “whole-unit-test” style is more efficient than code with “unit-by-unit test” style, it is not a significant difference because of p-value which is greater than 0.05. Finally, in “code understandability” category, although efficient code is sometimes hard to understand for rest of

team members, its efficiency is significant better than code written by students who write redundant code but is easy to understand for their teammates.

Table 21. Impact of programming styles subscale on coding performance

Programming Style	Project	Index	Running time(second)	P-value(one-tail)
Alone	1	Program alone	Mean 7.37E-04, Std. 0.00030, n=31	4.16E-04
		Program with a group	Mean 2.06E-04, Std. 0.0022, n=222	
	2	Program alone	Mean 1.41E-03, Std. 0.000904, n=32	2.45E-05
		Program with a group	Mean 4.5E-04, Std. 0.0023, n=223	
	3	Program alone	Mean 0.0239, Std. 0.0148, n=27	0.0125
		Program with a group	Mean 0.0039, Std. 0.122, n=209	
	4	Program alone	Mean 0.1246, Std. 0.066, n=22	1.56E-04
		Program with a group	Mean 0.047, Std. 0.172, n=134	
	5	Program alone	Mean 0.0052, Std. 0.0059, n=23	4.02E-04
		Program with a group	Mean 0.0045, Std. 0.179, n=166	
	6	Program alone	Mean 0.502, Std. 0.101, n=19	0.0382
		Program with a group	Mean 0.260, Std. 1.55, n=135	
	7	Program alone	Mean 0.683, Std. 0.207, n=16	0.3311
		Program with a group	Mean 0.656, Std. 0.392, n=125	
Continuous	1	Program continuously	Mean 7.12E-04, Std. 0.0025, n=144	0.3498
		Program intermittently	Mean 6.18E-04, Std. 0.0014, n=106	
	2	Program continuously	Mean 0.0010, Std. 0.0020, n=147	0.0169
		Program intermittently	Mean 0.0016, Std. 0.0025, n=108	
	3	Program continuously	Mean 0.00548, Std. 0.0406, n=137	0.4904
		Program intermittently	Mean 0.0054, Std. 0.0166, n=100	
	4	Program continuously	Mean 0.0781, Std. 0.1036, n=97	0.0017
		Program intermittently	Mean 0.1712, Std. 0.2205, n=59	
	5	Program continuously	Mean 0.0040, Std. 0.0062, n=109	0.1224
		Program intermittently	Mean 0.0068, Std. 0.0211, n=80	

	6	Program continuously	Mean 0.4731, Std. 1.8545, n=91	0.1462
		Program intermittently	Mean 0.2669, Std. 0.0862, n=63	
	7	Program continuously	Mean 0.6606, Std. 0.4116, n=86	0.2048
		Program intermittently	Mean 0.7110, Std. 0.3099, n=55	
Open Source	1	Open Source	Mean 5.21E-04, Std. 0.0013, n=152	0.0468
		Close Source	Mean 0.0010, Std. 0.0029, n=101	
	2	Open Source	Mean 0.0012, Std. 0.0022, n=152	0.0363
		Close Source	Mean0.0017, Std. 0.0022, n=103	
	3	Open Source	Mean 0.0030, Std. 0.0123, n=136	0.0487
		Close Source	Mean 0.021, Std. 0.1071, n=101	
	4	Open Source	Mean 0.1065, Std. 0.1739, n=91	0.0059
		Close Source	Mean 0.1887, Std. 0.2131, n=65	
	5	Open Source	Mean 0.0060, Std. 0.0183, n=109	0.1610
		Close Source	Mean 0.0041, Std. 0.0066, n=80	
	6	Open Source	Mean 0.4858, Std. 1.9506, n=82	0.1610
		Close Source	Mean 0.2781, Std. 0.1560, n=72	
	7	Open Source	Mean 0.6943, Std. 0.4689, n=79	0.0172
		Close Source	Mean 0.8262, Std. 0.2508, n=62	
Visual	1	Visual enviro	Mean 7.29E-04, Std. 0.0022, n=211	0.0438
		Text enviro	Mean 3.86E-04, Std. 8.3E-04, n=42	
	2	Visual enviro	Mean 0.0014, Std. 0.0023, n=215	0.0580
		Text enviro	Mean 9.08E-04, Std. 0.0015, n=40	
	3	Visual enviro	Mean 0.0058, Std. 0.0352, n=198	0.0150
		Text enviro	Mean 3.65E-04, Std. 8.47E-04, n=38	
	4	Visual enviro	Mean 0.1201, Std. 0.1755, n=129	0.0508
		Text enviro	Mean 0.0827, Std. 0.0851, n=27	
	5	Visual enviro	Mean 0.0053, Std. 0.0160, n=152	0.0136
		Text enviro	Mean 0.0023, Std. 0.0022, n=37	
	6	Visual enviro	Mean 0.2741, Std. 0.1399, n=121	0.1624
		Text enviro	Mean 0.8089, Std. 3.0710, n=33	
	7	Visual enviro	Mean 0.6862, Std. 0.4083, n=115	1.44E-04
		Text enviro	Mean 0.4727, Std. 0.2100, n=16	
Units	1	Unit by unit	Mean 6.88E-04, Std. 0.0022, n=165	0.0137
		All units	Mean 0.0014, Std. 0.0024, n=88	
	2	Unit by unit	Mean 0.0013, Std. 0.0023, n=169	0.0335
		All units	Mean 0.0018, Std. 0.0022, n=86	
	3	Unit by unit	Mean 0.0064, Std. 0.0388, n=159	0.1994
		All units	Mean 0.0035, Std. 0.0130, n=78	
	4	Unit by unit	Mean 0.1035, Std. 0.1365, n=102	0.0043
		All units	Mean 0.2070, Std. 0.2632, n=54	
	5	Unit by unit	Mean 0.0053, Std. 0.0173, n=121	0.0063
		All units	Mean 0.0640, Std. 0.1885, n=68	
	6	Unit by unit	Mean 0.4501, Std. 1.7937, n=97	0.1844
		All units	Mean 0.2843, Std. 0.1740, n=57	
	7	Unit by unit	Mean 0.6545, Std. 0.2697, n=94	0.0210

		All units	Mean 0.8161, Std. 0.4979, n=47	
EF	1	Efficient	Mean 1.68E-04, Std. 2.67E-04, n=56	5.02E-04
		Inefficient	Mean 6.7E-04, Std. 0.0020, n=197	
	2	Efficient	Mean 1.65E-04, Std. 1.81E-04, n=58	3.19E-12
		Inefficient	Mean 0.0014, Std. 0.0022, n=197	
	3	Efficient	Mean 3.47E-04, Std. 5.85E-04, n=52	0.0181
		Inefficient	Mean 0.0060, Std. 0.036, n=185	
	4	Efficient	Mean 0.033, Std. 0.0186, n=30	8.43E-09
		Inefficient	Mean 0.1177, Std. 0.1540, n=126	
	5	Efficient	Mean 0.0029, Std. 0.0037, n=41	0.0429
		Inefficient	Mean 0.0053, Std. 0.0158, n=148	
	6	Efficient	Mean 0.1518, Std. 0.0480, n=32	0.0329
		Inefficient	Mean 0.4215, Std. 1.6023, n=122	
	7	Efficient	Mean 0.1518, Std. 0.1908, n=32	0.0329
		Inefficient	Mean 0.4215, Std. 0.3958, n=122	

The academic performance analysis is difference from the analysis of the coding performance because more consideration needs to be added in rubrics such as specific iteration usability. In the “programming alone” category, we can see, in Table 22, that the code written by the “alone” style is significantly more efficient than code with the “group” styles because p-value in 4 assignment is less than 0.05.

Since programmers with the “continuous” style can focus on the code in a consecutive period, the consistency of the code style can be perfectly maintained. Hence we propose that the grade from the “continuous programming” style is significantly higher than the grade from the “intermittent programming” style and the p-value in Table 22 supports our hypothesis. Similarly, in the “Open Source maintenance” category, the grade from the “Open Source” type is significantly higher than the grade from the “Closed Source” type. The only exception happened to the project 1, but it does not affect our hypothesis that grades from students with the “Open Source” style is significantly higher than grades from students with the “Closed Source” style.

Another category is the “visual programming” context. After the analysis, we can see the grades from students who used to writing codes in a text environment is significantly higher than grade with the “visual programming context”. Unfortunately, in the “unit-test” category, we did not find any valid p-value which can support a significant difference. Finally, in the “code efficiency” category, students with the “efficient code” style achieve a higher grade than student with the “redundant code” style.

Table 22. Impact of programming styles subscale on academic performance

Programming Style	Project	Index	Grade	P-value(one-tail)
Alone	1	Program alone	Mean 94.4, Std. 7.14, n=35	5.79E-05
		Program with a group	Mean 88.12, Std. 15.63, n=252	
	2	Program alone	Mean 86.04, Std. 86.04, n=32	0.00335
		Program with a group	Mean 78.58, Std. 78.58, n=253	
	3	Program alone	Mean 83.60, Std. 21.99, n=31	0.0179
		Program with a group	Mean 83.67, Std. 21.48, n=228	
	4	Program alone	Mean 92.93, Std. 11.78, n=29	3.88E-04
		Program with a group	Mean 83.82, Std. 19.69, n=227	
Continuous	1	Program continuously	Mean 90.03, Std. 13.57, n=162	0.0153
		Program intermittently	Mean 85.99, Std. 17.02, n=125	
	2	Program continuously	Mean 82.16, Std. 20.05, n=163	0.0071
		Program intermittently	Mean 75.79, Std. 22.66, n=123	
	3	Program continuously	Mean 85.87, Std. 20.07, n=148	0.0309
		Program intermittently	Mean 80.73, Std. 23.02, n=111	
	4	Program continuously	Mean 87.85, Std. 17.40, n=145	7.71E-04
		Program intermittently	Mean 79.91, Std. 21.19, n=111	
	1	Open Source	Mean 88.49, Std. 15.29, n=170	0.3882
		Close Source	Mean 87.97, Std. 15.32, n=117	
	2	Open Source	Mean 84.80, Std. 16.07, n=164	0.0037
		Close Source	Mean 78.45, Std. 21.81, n=121	

Open Source	3	Open Source	Mean 87.89, Std. 17.85, n=146	0.0138
		Close Source	Mean 82.42, Std. 20.93, n=112	
	4	Open Source	Mean 87.94, Std. 15.56, n=147	0.0215
		Close Source	Mean 83.35, Std. 19.62, n=112	
Visual	1	Visual enviro	Mean 87.43, Std. 15.99, n=240	0.0023
		Text enviro	Mean 92.60, Std. 9.93, n=47	
	2	Visual enviro	Mean 78.34, Std. 21.77, n=239	0.0179
		Text enviro	Mean 85.02, Std. 18.89, n=46	
	3	Visual enviro	Mean 83.59, Std. 21.28, n=216	0.0081
		Text enviro	Mean 90.09, Std. 14.56, n=43	
	4	Visual enviro	Mean 83.95, Std. 19.53, n=213	3.84E-04
		Text enviro	Mean 91.33, Std. 10.86, n=43	
Unit	1	Unit by unit	Mean 88.13, Std. 14.99, n=186	0.42
		All units	Mean 88.52, Std. 15.86, n=101	
	2	Unit by unit	Mean 79.11, Std. 21.86, n=187	0.37
		All units	Mean 80.00, Std. 20.73, n=98	
	3	Unit by unit	Mean 84.35, Std. 21.12, n=173	0.23
		All units	Mean 82.28, Std. 22.27, n=86	
	4	Unit by unit	Mean 83.36, Std. 20.34, n=170	0.10
		All units	Mean 86.48, Std. 17.65, n=86	
EF	1	Efficient	Mean 92.78, Std. 10.79, n=65	0.005
		Inefficient	Mean 88.48, Std. 15.36, n=222	
	2	Efficient	Mean 90.24, Std. 10.45, n=63	7.99E-08
		Inefficient	Mean 79.73, Std. 21.18, n=222	
	3	Efficient	Mean 89.66, Std. 15.68, n=58	0.012
		Inefficient	Mean 83.87, Std. 21.05, n=201	
	4	Efficient	Mean 84.41, Std. 19.66, n=55	0.500
		Inefficient	Mean 84.41, Std. 19.51, n=201	

In Tables 23-26, the acronym has been explained: (1) AG' -- Programming alone or programming with a group; (2) CI' -- Programming continuously or programming intermittently; (3) OC' -- Maintenance in an open source way or closed source way; (4) VT' -- Programming in a visual context or in a text context; (5) US' -- Test codes unit by unit or test whole codes at one time; (6) EF' -- Write efficient code but hard to understand or write redundant code but easy to understand.

We analyzed the linear relationship between programming styles and the coding performance in Table 23. In the “programming alone” column, the linear coefficient is slightly less than 0.7 in project 1 and 2. But for the rest of project the coding performance is linearly dependent to the “programming alone” style because the linear coefficient is greater than 0.7. In the “continuous programming” style, since all p-values is negative numbers, we make a conclusion that the coding performance is positively linearly dependent to the “intermittent programming” style instead of the “continuous programming” style.

In the “Open-Source maintenance” styles, we did not find any p-value is greater than 0.7 or less than -0.7. Hence there is no strong linear relationship between this programming style and the coding performance. For the “visual programming context” style, the data in Table 23 showed that the strong linear relationship can be found only in project 1, 2, 3 and 5. The results from project 3, 6, and 7 showed that the fourth programming style in our research is not statistically linearly dependent to the coding performance. Based on the analysis above, we make a conclusion that there is no significant linear dependence between the coding performance and the programming context style.

For the fifth style: Unit-Test, we found that the all linear coefficients are less than -0.7 which means the running time rises up with the increment of statistical values in the “testing whole units at one time” style. Finally, in the “efficient-code” style, the most of experimental results are good to support our hypothesis that the running time increases when statistical values in the

"redundant code" style rise. We also notice that in project 6 and 7 the coefficient value is slightly greater than -0.7 with the reason of the high complexity of projects themselves.

Table 23. Correlation of influence of programming styles subscale on coding performance

Styles / Project	AG'	CI'	OC'	VT'	US'	EF'
1	0.6401	-0.7664	-0.0479	-0.7761	0.7847	0.7624
2	0.6885	-0.7572	-0.0194	-0.7780	0.8272	0.7882
3	0.8604	-0.8369	-0.0472	-0.7428	0.8983	0.7785
4	0.9074	-0.8010	-0.0305	-0.6341	0.9155	0.7185
5	0.9012	-0.8738	0.0169	-0.7661	0.9394	0.7909
6	0.8284	-0.7161	0.1984	-0.5907	0.8972	0.6291
7	0.9187	-0.7732	-0.0087	-0.5814	0.9749	0.6802

In Table 24 the students' grade of each project was analyzed to explore a linear dependence with the programming styles. Except the third programming style ("Open Source"), we found that the grade of each project is significantly linearly dependent to the positive aspect of each category.

Table 24. Correlation of influence of programming styles on academic performance

Styles / Assignment	AG'	CI'	OC'	VT'	US'	EF'
1	0.9659	0.9138	-0.0086	0.8929	0.9611	0.8935
2	0.9789	0.9104	-0.0185	0.8985	0.9707	0.9078
3	0.9783	0.9362	-0.0579	0.8767	0.9800	0.8845
4	0.9389	0.9208	-0.1005	0.8531	0.9714	0.8558

In order to help researchers and educators understand the impact of each factor of the programming styles, we also show a prediction model in Table 25. Totally there are 59 students who correctly solved 7 projects and R-square is 0.9. Besides, in Figure 23-28 all points are randomly dispersed around the horizontal axis. Hence, we make a conclusion the regression model fits our data.

Table 25. Regression of influence of programming styles subscale on coding performance

Project		Coefficients	P-value	R Square	N
Performance	Intercept	0.078	0.14	0.90	59
	AG'	-0.0023	0.133		
	CI'	0.008	0.025		
	OC'	4.24E-04	0.76		
	VT'	0.007	0.02		
	US'	0.0022	0.47		
	EF'	0.018	1.39E-06		
Regression Equation	-0.0023*AG'+0.008*CI'4.24E-04*OC'+0.007*VT'+0.0022*US'+0.018*EF'+0.078				

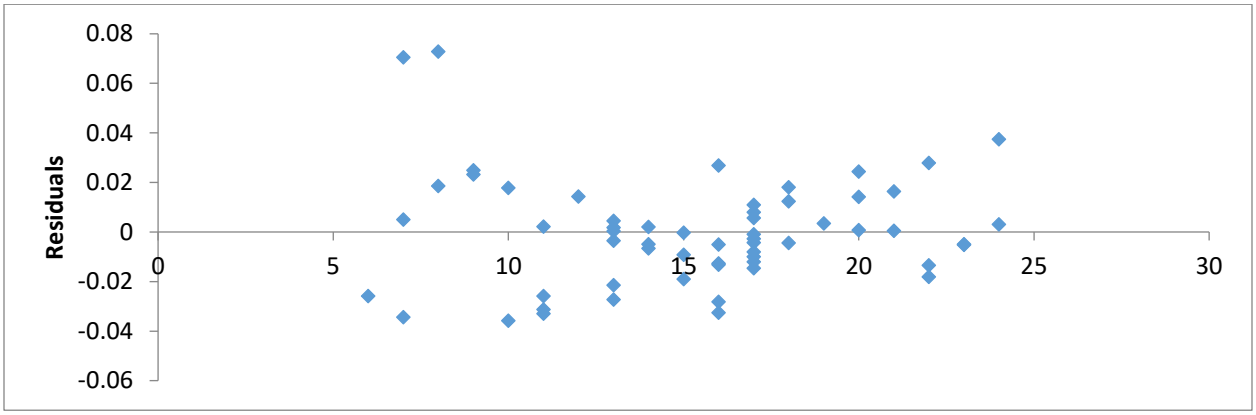

Figure 23. Alone vs. Group residual plot for coding performance

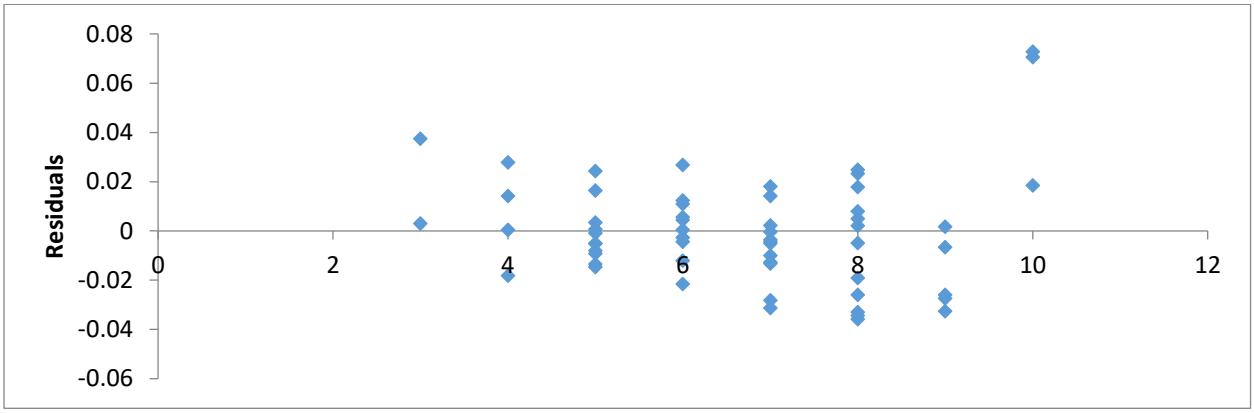

Figure 24. Continuous vs. Intermittent residual plot for coding performance

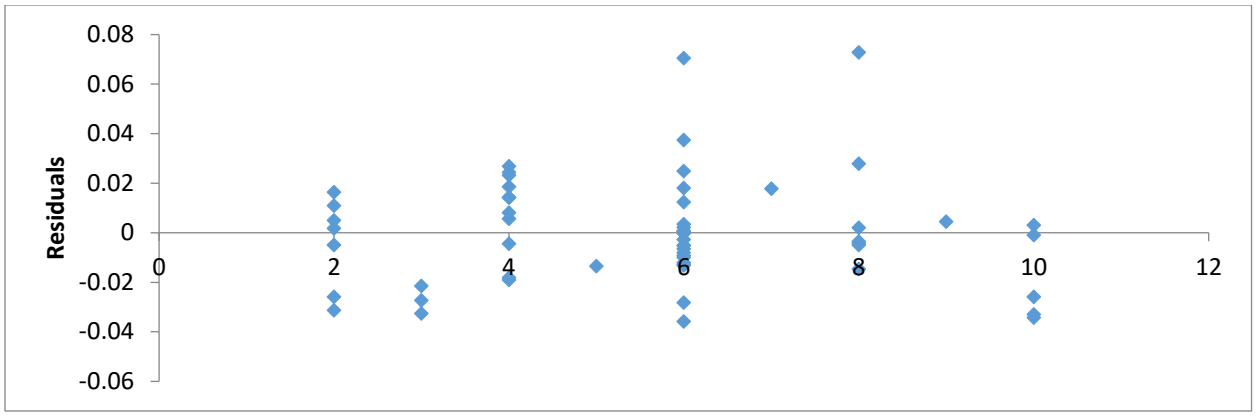

Figure 25. Open Source vs. Closed Source residual plot for coding performance

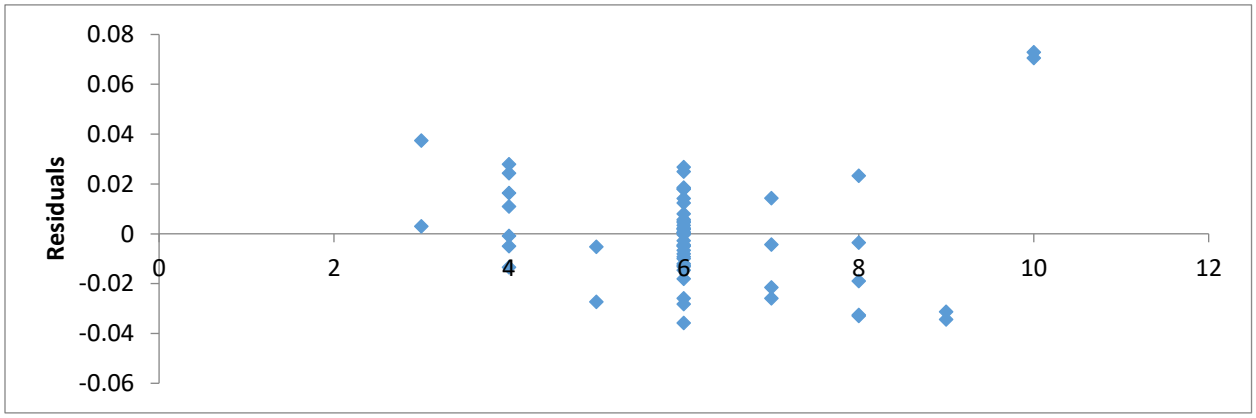

Figure 26. Visual vs. Text residual plot for coding performance

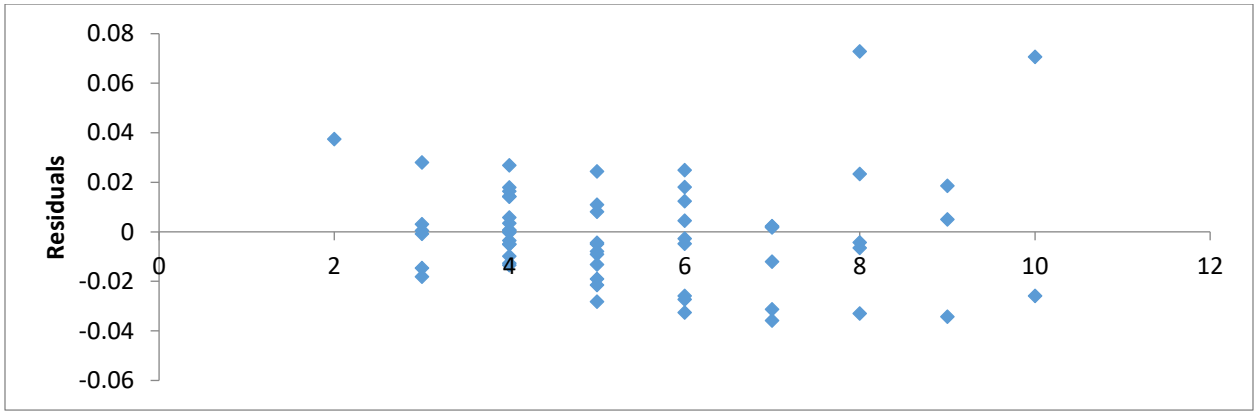

Figure 27. Single Unit vs. Whole Units residual plot for coding performance

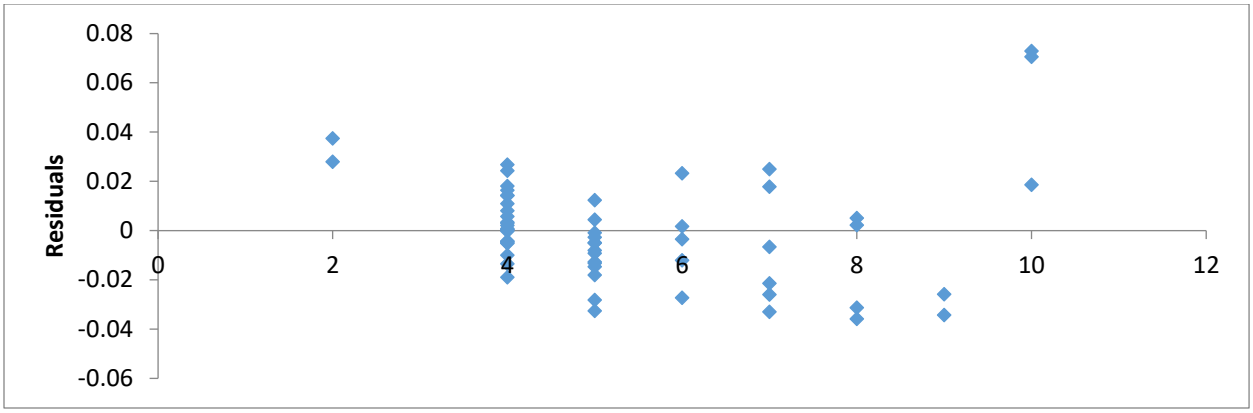

Figure 28. Efficient vs. Inefficient residual plot for coding performance

Also, in Table 26 we present a regression model that predicts the impact of factors in each category of programming styles on grades. To verify whether our model fits the data or not, we not only check the R-square but also discuss the points distribution in residual plots in Figure 29-34. Although in Figure 32 some points at the beginning section are located below the x-axis, it does not affect the whole distribution in our experimental data. Hence, the regression model presented in Table 26 is acceptable to predict participants' academic performance.

Table 26.Regression of influence of programming styles subscale on academic performance

Project		Coefficients	P-value	R Square	N
Grade	Intercept	0.55	1.38E-24	0.82	218
	AG'	-0.0028	0.04		
	CI'	0.006	0.11		
	OC'	5.03E-04	0.71		
	VT'	0.019	5.83E-07		
	US'	0.0086	0.002		
	EF'	0.0087	7.32E-04		
Regression Equation	-0.0028*AG'+0.006*CI'+5.03E-04*OC'+0.019*VT'+0.0086*US'+0.0087*EF'+0.55				

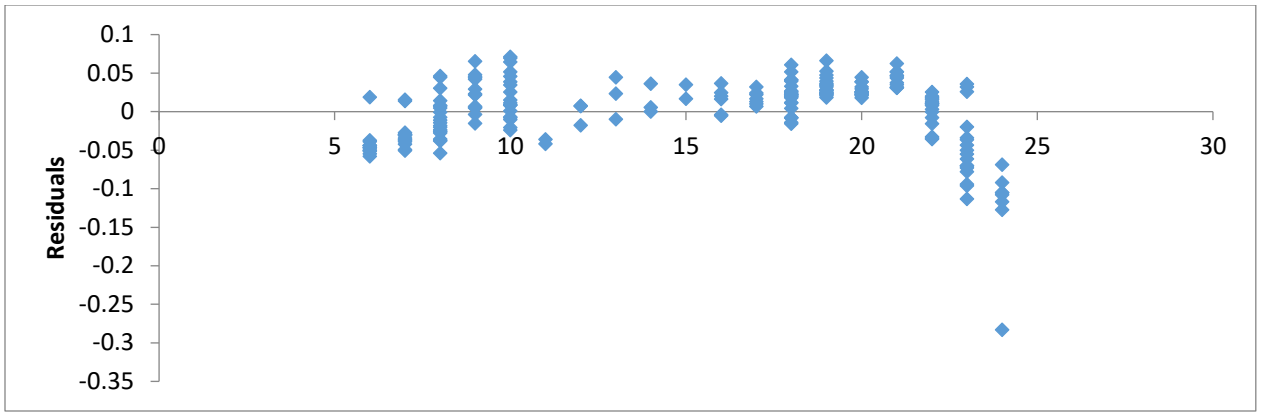

Figure 29. Alone vs. Group residual plot for academic performance

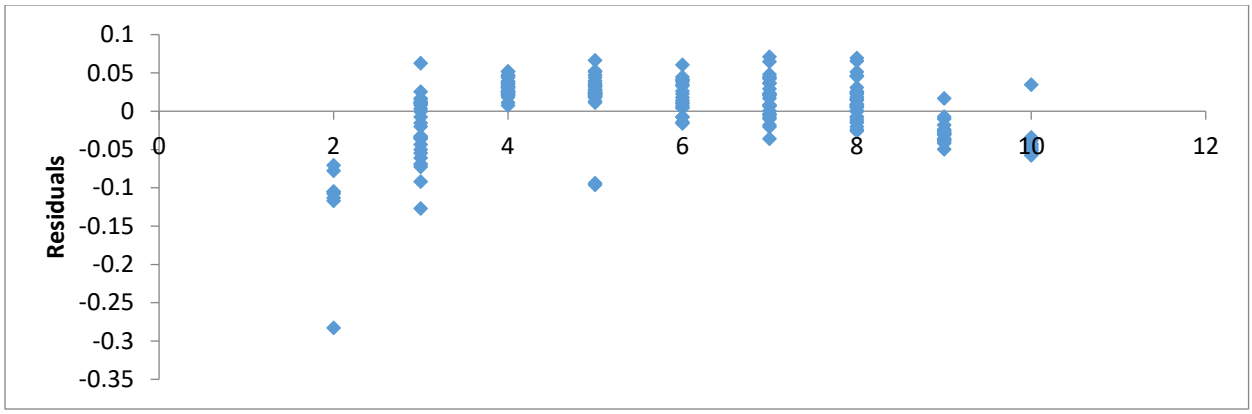

Figure 30. Continuous vs. Intermittent residual plot for academic performance

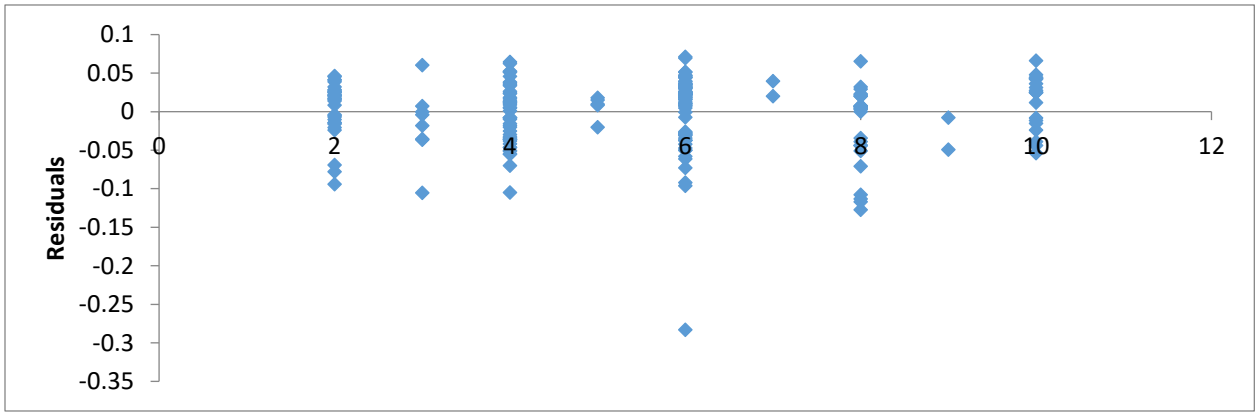

Figure 31. Open Source vs. Closed Source residual plot for academic performance

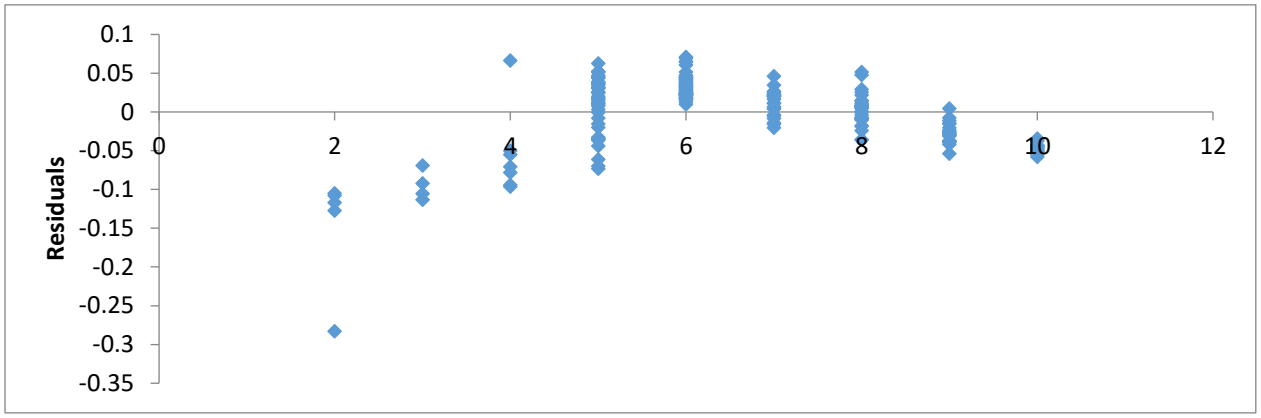

Figure 32. Visual vs. Text residual plot for academic performance

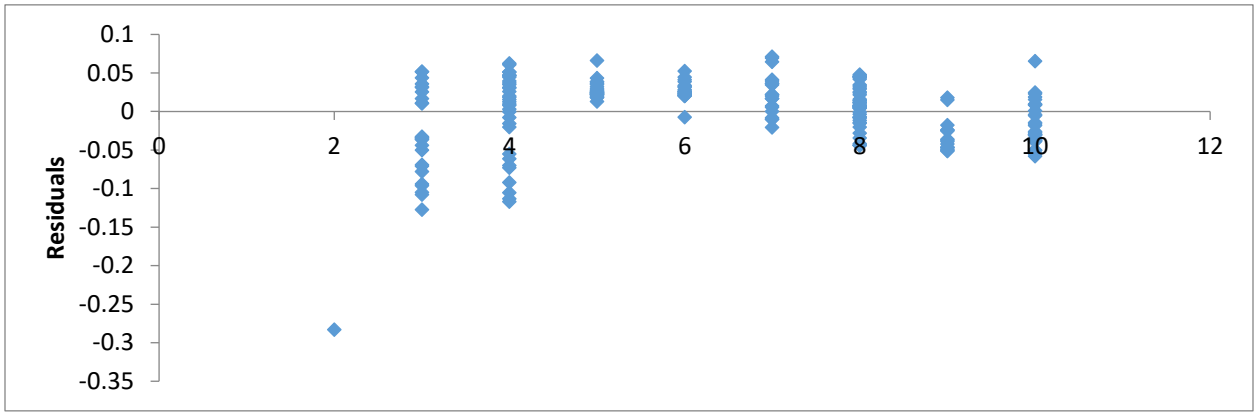

Figure 33. Single Unit vs. Whole Units residual plot for academic performance

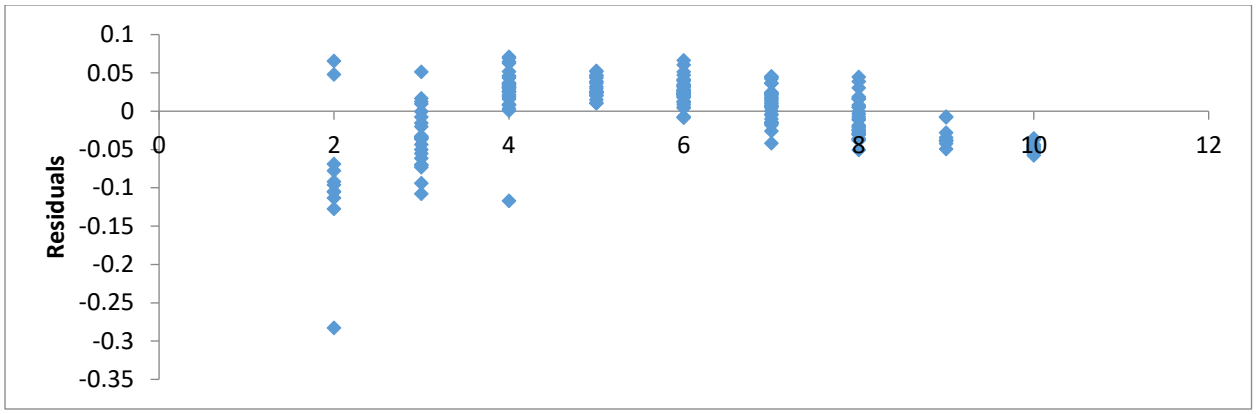

Figure 34. Efficient vs. Inefficient residual plot for academic performance

4.5.4 The relationship among human factors

In this section, we explore the correlation relationship among three human factors: Personality, Programming Styles and Programming Attitude. Because some research papers explored the relationship between partial factors of the programming attitudes and the personality. In our research we conducted this experiment with 328 participants. The data in Table 27 tell us that there is no significant correlation relationship in between because all coefficient values are less than 0.7.

Table 27. Correlation between personality and attitude

Personality / Attitude	EI'	SN'	TF'	JP'
CNC'	-0.1651	0.0512	0.1613	0.2091
ATNAT'	0.0332	0.0677	0.1303	0.0971
MDFD'	-0.0383	0.1416	0.1164	0.1193
UFUL'	-0.2257	0.0558	0.2130	0.1258
EMIM'	-0.1822	0.0745	0.2261	0.2447

Another pair of human factor relationship in Table 28 is the personality and programming styles. In the " EI' " column, we only found one exception: there is no significant correlation between the "Favorite World" personality and the "Unit-Test" programming styles. Secondly, in

"Decision" personality, no results can support the "Open Source" and "Unit-Test" styles have the strong correlation relationship with the "Thinking" or "Feeling" characters. Finally, for "Structure" personality, we found that the "Visual" and "Unit-Test" styles do not correspondingly change with the "Judging" or "Perceiving" characters.

Table 28. Correlation between personality and programming styles

Personality / Styles	EI'	SN'	TF'	JP'
Alone	-0.80	-0.83	0.85	0.80
Continuous	-0.86	-0.86	0.85	0.87
Open Source	0.79	0.78	0.04	-0.78
Visual	0.81	0.79	-0.81	-0.07
Units	-0.04	-0.90	0.03	0.15
Efficient	-0.87	-0.83	0.84	0.85

In the analysis of relationship between programming attitudes and programming styles in Table 29, under each category of attitudes, we identified some programming styles, which are not related to attitudes. They are listed in the following:

(1) In the "Confidence" column, the "Visual" and "Unit-Test" programming styles do not have a statistically strong correlation relationship. And there is a negative correlation relationship between the "Open Source" style and the "Confidence" attitude.

(2) In the "Success" column, the "Open Source" and "Visual" styles are not linearly dependent to the "Success" attitude. But the rest of programming styles are strongly linearly dependent to it.

(3) In the " MDFD' "column, the "Continuous programming" and "Unit-Test" styles are not linearly dependent to the "male-domain" attitude. And the negative correlation relationship exists between the "computer science as a male domain" attitude and "Open Source"/ "Visual programming" styles.

(4) In the “ UFUL’ ” column, the “Open Source” and “Visual programming” styles are not linearly dependent on the “Useful” attitude.

(5) In the “ EMIM’ ” column, the “programming alone” and “Visual programming” styles are not linearly dependent on the “Effective” attitude.

Table 29. Correlation between attitude and programming styles

Attitude / Styles	CNC’	ATNAT’	MDFD’	UFUL’	EMIM’
Alone	0.87	0.75	0.91	0.97	0.49
Continuous	0.91	0.80	-0.43	0.81	0.88
Open Source	-0.72	-0.66	-0.75	0.11	0.76
Visual	-0.59	-0.39	-0.88	-0.61	-0.23
Units	-0.65	0.79	0.59	0.80	0.86
Efficient	0.87	0.89	0.88	0.95	0.91

Chapter 5 Conclusion

In this Chapter, we present the answers of the hypotheses proposed in Charter 4, discuss our results and list the future work.

5.1 Conclusions of hypotheses

Although the comprehensive analysis has been explained in Chapter 4, it is necessary to gather all conclusions at one time to clearly show them to researchers.

(1) which factors of the personality significantly play a positive/negative role in coding performance.

The following factors of personality significantly play a positive role in the code performance: Introversion, Intuition, Thinking and Perceiving characters, while the following factors of the personality significantly play a negative role in the code performance: Extraversion, Sensing, Feeling and Judging.

(2) which factors of the personality significantly play a positive/negative role in the academic performance.

The following factors of the personality significantly play a positive role in the academic performance: Introversion, Sensing, Thinking and Judging characters, while the following factors of the personality significantly play a negative role in the academic performance: Extraversion, Intuition, Feeling and Perceiving.

(3) which factors of programming attitudes significantly play a positive/negative role in the coding performance.

The following factors of programming attitudes significantly play a positive role in the code performance: Confidence, Success, Male domain, Usefulness and Effectiveness characters, while the following factors of programming attitudes significantly play a negative role in the code performance: Non-Confidence, Non-Success, Female domain, Non-usefulness and Ineffectiveness.

(4) which factors of programming attitudes significantly play a positive/negative role in the academic performance.

The following factors of programming attitudes significantly play a positive role in the academic performance: Confidence, Success, Male domain, Usefulness and Effectiveness characters, while the following factors of programming attitudes significantly play a negative role in the academic performance: Non-Confidence, Non-Success, Female domain, Non-usefulness and Ineffectiveness.

(5) Whether the programming styles can be verified.

Nine subscales of programming styles were developed and the results showed that 6 of 9 subscales were verified with Cronbach's alpha values which are greater than 0.7: (1) Programming Alone; (2) Continuous programming; (3) Open source maintenance; (4) Visual programming context; (5) Unit test; and (6) Efficient code. The rest of three styles failed to be verified: (1) Discussion in the office; (2) Whiteboard usability; and (3) Automatic synchronization of code.

(6) which factors of programming styles significantly play a positive/negative role in the coding performance.

The following factors of programming styles significantly play a positive role in the code performance: Group programming, Continuous programming, Open source maintenance, Text programming context, Unit Test and Efficient Code styles, while the following factors of programming attitudes significantly play a negative role in the code performance: Programming alone, Intermittent programming, Closed Source maintenance, Visual programming context, Whole-Unit-Test and Redundant Code styles.

(7) which factors of programming styles significantly play a positive/negative role in the academic performance.

The following factors of programming styles significantly play a positive role in the academic performance: Programming alone, Continuous programming, Open source maintenance, Text programming context and Efficient Code styles, while the following factors of programming attitudes significantly play a negative role in the academic performance: Group programming, Intermittent programming, Closed source maintenance, Visual programming context, whole-unit-test and redundant code. We also identified the "Unit-Test" style as a special one because it does not have significant both negative and positive roles in the academic performance

(8) whether there exists a strong linear relationship among factors of the personality and the coding performance or not.

The following factors of the personality are significantly linear dependent on the code performance: Introversion, Sensing, Thinking and Judging characters, while the following factors of the personality are not significantly linear dependent on the code performance: Extraversion, Intuition, Feeling and Perceiving.

(9) whether there exists a strong linear relationship among factors of the personality and the academic performance or not.

The following factors of the personality are significantly linear dependent on the academic performance: Introversion, Sensing, Thinking and Judging characters, while the following factors of the personality are significantly linear dependent on academic performance: Extraversion, Intuition, Feeling and Perceiving.

(10) whether there exists a strong linear relationship among factors of the programming attitudes and the coding performance or not.

The following factors of programming attitude are significantly linear dependent on the coding performance: Confidence, Success, Usefulness and Effectiveness characters, while the following factors of programming attitude are not significantly linear dependent on the coding performance: computer science as a male domain.

(11) whether there exists a strong linear relationship among factors of programming attitude and the academic performance or not.

The following factors of programming attitude are significantly linear dependent on the academic performance: Confidence, Success, Usefulness and Effectiveness characters, while the following factors of programming attitude are not significantly linear dependent on the academic performance: computer science as a male domain.

(12) whether there exists a strong linear relationship among factors of programming styles and the coding performance or not.

The following factors of programming styles are significantly linear dependent on the code performance: Programming alone, Continuous programming, Text programming context, Unit Test and Efficient code characters, while the following factors of programming attitudes are not significantly linear dependent on the code performance: Programming alone, Intermittent programming, Closed source maintenance, Visual programming context, Whole-Unit-Test and Redundant code. Finally, we identified the "Open Source maintenance" style as a special one because there is neither a positive nor a negative role in the code performance.

(13) whether there exists a strong linear relationship among factors of programming styles and the academic performance or not.

The following factors of programming styles are significantly linear dependent on the academic performance: Programming alone, Continuous programming, Text programming context, Unit Test and Efficient code characters, while the following factors of programming styles are not significantly linear dependent on the academic performance: Programming alone, Intermittent programming, Closed source maintenance, Visual programming context, Whole-Unit-Test and Redundant code. Finally, we identified the "Open Source maintenance" style as a special one because there is neither a positive nor a negative role in the code performance.

(14) whether the impact of personality factors on the coding performance can be predicted or not.

The regression model has been conducted on the impact of the personality factors on the code performance: coding performance = -0.007*E+8.4E-04*S+0.032T-7.4E-04*J+0.38

(15) whether the impact of the personality factors on the academic performance can be predicted or not.

The regression model cannot be conducted on the impact of the personality factors on the academic performance since the U-shape model appeared in the residual plots of four categories of personality.

(16) whether the impact of programming attitudes on the code performance can be predicted or not.

Although the R square is less than 0.7, the points of five categories of programming attitudes on residual plots are perfectly dispersed around the horizontal axis. Hence the regression model can be trusted: 0.0045*C-3.4E-04*AT+2.52E-04*MD+0.0013*UF-0.0025*EM+0.076.

(17) whether the impact of programming attitudes on the academic performance can be predicted or not.

The R-square is 0.98, the points of five categories of programming attitudes on the residual plot are perfectly dispersed around the horizontal axis. Hence the regression model can be trusted: 0.0039*C-0.0015*AT-1.2E-05*MD-6.1E-04*UF+0.009*EM+0.24

(18) whether the impact of factors of programming styles on the code performance can be predicted or not.

The R-square is 0.90, the points of six categories of programming styles on the residual plot are perfectly dispersed around the horizontal axis. Hence the regression model can be trusted: -0.0023*A+0.008*C+4.24E-04*O+0.007*V+0.0022*U+0.018*E+0.078

(19) whether the impact of programming styles on the academic performance can be predicted or not.

The R-square is 0.82, the points of six categories of programming styles on all residual plots are perfectly dispersed around the horizontal axis. Hence the regression model can be trusted: -0.0028*A+0.006*C+5.03E-04*O+0.019*V+0.0086*U+0.0087*E+0.55

(20) whether there exists a strong linear relationship between the personality and programming styles or not.

For the "Favorite World" category of the personality, all subscales (except the "Unit Test" type) of programming styles are linearly dependent on the "Extraversion" type; for the "Information" category of the personality, all subscales of programming styles are linearly dependent on the "Sensing" personality; for the "Decision" category of the personality, all subscales, except the "Open Source" and "Unit Test" styles, of programming styles are linearly dependent on the "Thinking" personality and for the "Structure" category of the personality, all subscales, except the "Visual context" and "Unit Test", of programming styles are linearly dependent on the "Judging" personality.

(21) whether there exists a strong linear relationship between programming attitudes and programming styles or not.

For the "Confidence" programming attitude, all subscales, except the "Visual programming context" and "Unit Test", of programming styles are linearly dependent on it; all subscales of programming styles, except the "Visual programming context" and "Open Source" styles, are linearly dependent on the "Success" attitude; all subscales, except the "Continuous programming" and "Unit Test" styles, of programming styles are linearly dependent on the "computer science as a male domain" attitude; all subscales, except the "Visual context" and "Open Source" styles, of programming styles are linearly dependent on the "Judging"

personality; and all subscales, except the “Programming alone” and “Visual context” styles, of programming styles are linear dependent on the “Effective” attitude.

(22) whether there exists a strong linear relationship between the personality and programming attitudes or not.

Unfortunately, we did not find any significant linear relationship between any subscale of the personality and programming attitudes.

5.2 Contribution

Our research work explores the influence of human factors on the programming performance. Coding and Computer Programming have become an indispensable part of every company (e.g. even companies that do not specialize in creating computing technology) and supports the technologies that change our way of life. We need methodologies to help identify what will aid programmers in being more effective in this profession. The results of this study can provide guidelines for new computing programmers and their instructors to provide better assignments and classroom instruction based on this work in personality, attitudes and programming styles. The contributions can be summarized as follows:

(1) The contemporary programming styles are successfully updated and verified with Cronbach Alpha. The previous research work explored the relationship between personality styles and programming styles based on pre object-oriented programming styles (Vessey, 1985). Our research work updates the programming styles after 2006, verified them with Cronbach Alpha, and extended the experiment to also contain object-oriented programming styles.

(2) Based on the analysis of the relationship between programming styles and performance, guidelines are provided for positive and negative impact of each subscale of programming styles.

(3) The prediction model represented by regression is also provided between performance and three human factors: MBTI personality, programming styles and programming attitude.

(4) The linear dependence between performance and three human factors has been statistically explored.

5.3 Future Work

In our research, we systematically conducted and analyzed the empirical study: the influence of human factors on the performance: personality, programming attitude and programming styles with a variety of background of students. Since most participants are novices or entry-level programmers, our experiment still needs to be improved (e.g. inviting programming experts or senior software engineers).

Secondly, we did not consider the impact of projects' complexity on program performance. Although the last two projects are harder than the first five projects, overall there is no significant difference in projects complexity. For the following work, we will intend to create projects based on the complexity gradient.

Finally, the programming style still needs to be polished carefully. Currently, the survey of programming styles still needs to be sent back to the same population and collect it again. The

planned future activity is to continue experimentation until all questions are verified with Cronbach's alpha.

References

[1] Iris, V. Expertise in debugging computer programs: A process analysis. International Journal of Man-Machine Studies, 23(5), 459-494.

[2] Ko, A. J. and Uttl, B. Individual Differences in Program Comprehension Strategies in Unfamiliar Programming Systems. International Conference on Program Comprehension, p. 175, 11th IEEE International Workshop on Program Comprehension (IWPC'03), 2003.

[3] Cox, A., Fisher, M. Programming style: Influences, factors, and elements. Proceedings of the second international conferences on advance in Computer-Human Interactions ACHI'09, pp. 82-89.

[4] Allison, E. T., Brian, D., Oliver, S. Toward a validated computing attitudes survey. Proceedings of the ninth annual international conference on International computing education research. pp. 135-142.

[5] Eric, W., Laurie, W., Kai, Y., Carol, M. Computer Science Attitude Survey. Technical Report in North Carolina State University at Raleigh.

[6] Michael, D. B., Nhung, T. N., Christopher, J. L. C. Common method variance in NEO-FFI and IPIP personality measurement. In the 24rd Annual Conference of The Society for Industrial and Organizational Psychology, New Orleans, LA, 2009.

[7] Schmit, M.J., Ryan, A.M. The Big Five in Personnel Selection: Factor structure in applicant and nonapplicant populations. Journal of Applied Psychology, 1993(78): 966-974

[8] Cellar, D. F., Miller, M. L., Doverspike, D. D., Klawsky, J. D. (1996). Comparison of factor structures and criterion-related validity coefficients for two measures of personality based on the five factor model. Journal of Applied Psychology, 1996(81): 694-704.

[9] Lim, B-C., Ployhart, R. E. Assessing the Convergent and Discriminant Validity of Goldberg's International Personality Item Pool: A Multitrait-Multimethod Examination. Organizational Research Methods, 2006(9): 29-54.

[10] Costa, P. T., Jr., & McCrae, R. R.The NEO Personality Inventory manual. 1985, Odessa, FL: Psychological Assessment Resources.

[11] Costa, P. T., Jr., & McCrae, R. R. The NEO-PI/NEO-FFI manual supplement. 1989, Odessa, FL: Psychological Assessment Resources.

[12]Gown, A. J., Whiteman, M. C., Pattie, A. and Deary, I. J. Goldberg's 'IPIP' Big-Five factor markers: Internal consistency and concurrent validation in Scotland. Journal of Personality and Individual Differences. 2005(39) pp: 317-329.

[13] Zahra, K., Ahmad, B. D. Nasser, G. A. Influence of Personality on Programming Styles an Empirical Study. Journal of Information Technology Research. 2015(8): 38-56.

[14] Zahra, K., Ahmad, B. D. Nasser, G. A. Stefan, W. Links between the personalities, styles and performance in computer programming. Journal of Systems and Software. 2016(11): 228-241.

[15] Alessandra, D. D. C., David, G. Code review and personality: is performance linked to MBTI type? Technical Report at University of Newcastle. 2004.

[16] Edwards, J.A., Lanning, K. and Hooker, K. The MBTI and Social Information Processing: An Incremental Validity Study. Journal of Personality Assessment, 2002. 78(3): p. 432-450.

[17] Smither, R.D., The psychology of work and human performance. 3rd ed. 1998, New York: Longman. xviii, 590.

[18] 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.

[19] Devito, A.J., Review of Myers-Briggs Type Indicator, in The Ninth Mental measurements yearbook, J. Mitchell, Editor. 1985, Lincoln, Neb. pp. 1030-1032.

[20] Weinberg, G.M., The psychology of computer programming. 1971-1998, New York, Van Nostrand Reinhold. xv, 288.

[21] Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of paired programming in the introductory computer science course. Computer Science Education, 12 (3), 197-212.

[22] Myers, I. B., & McCaulley, M. H. (1985). Manual: A guide to the development and use of the Myers-Briggs Type Indicator. Palo Alto, CA: Consulting Psychologists Press.

[23] Myers, I. B., McCaulley, M. H., Quenk, N. L., & Hammer, A. L. (1998/2003). MBTI manual: A guide to the development and use of the Myers-Briggs Type Indicator (3rd ed.). Palo Alto, CA: Consulting Psychologists Press.

[24] Mendelsohn, G. A. (1965). Review of the Myers-Briggs Type Indicator. In O. K. Buros (Ed.) The sixth mental measurements yearbook. (pp. 321-322). Highland Park, NJ: Gryphon Press.

[25] Sundberg, N. D. (1965). Review of the Myers-Briggs Type Indicator. In O. K. Buros (Ed.) The sixth mental measurements yearbook. (pp. 322-325). Highland Park, NJ: Gryphon Press.

[26] Myers, I. B. (1962). Manual: The Myers-Briggs Type Indicator. Palo Alto, CA: Consulting Psychologists Press.

[27] Harvey, R. J. (1996). Reliability and validity. In A. L. Hammer (Ed.), MBTI applications (pp. 5-29). Palo Alto, CA: Consulting Psychologists Press.

[28] Fleenor, J. W. (2001). Review of the Myers-Briggs Type Indicator, Form M. In B. S. Plake & J. C. Impara (Eds.), The fourteenth mental measurements yearbook. (pp. 816-818). Lincoln, NB: The University of Nebraska Press.

[29] Bishop, C. C and Wheeler, D. D The Myers Briggs personality type and its relationship to computer programming. Journal of Research on Computing in Education, vol. 26, pp. 358-370, Spring 1994.

[30] Keirsey, D. Please Understand Me II. Del Mar, CA: Prometheus Nemesis Book Company, 1998.

[31] Williams, L. and Kessler, R. Pair Programming Illuminated, Addison Wesley, Reading,

Massachusetts, 2003.

[32] Williams, L., McDowell, C., Nagappan, N., Fernald, J. and Werner, L. "Building Pair Programming Knowledge through a Family of Experiments," proceedings of International Symposium on Empirical Software Engineering (ISESE), Rome, Italy, 2003, pp. 143-152.

[33] M. Collins. The Effects of Group Personality Composition on Project Team Performance: Operationalization and Outcomes. Dissertation. University of Tennessee.

[34] J. H. Bradley, F. J. Hebert. The effect of personality type on team performance. Journal of Management Development, Vol. 16 No. 5, 1997, pp. 337-353.

[35] Maio, G.R. and G. Haddock. The psychology of attitudes and attitude change. London: SAGE Publications Ltd. Doi: 10.4135/9781446214299. 2010.

[36] Zahra, K. and Stefan, W. The influence of personality on computer programming: a summary of a systematic literature review. http://dx.doi.org/10.18419/opus-3243. 2014.

[37] Andrew, H. and Moskal, M. Examining Science and Engineering Students' Attitudes Toward Computer Science. FIE'09 Proceedings of the 39th IEEE international conference on Frontiers in education conference. Pp. 1306-1311.

[38] Boehm, B.W., Software engineering economics. Prentice-Hall advances in computing

science and technology series. 1981, Englewood Cliffs, N.J.: Prentice-Hall. xxvii, 767.

[39] Whitley. B.E. The Relationship of Psychological Type to Computer Aptitude Attitudes and Behavior. Computers in Human Behavior. vol. 12 no. 3 pp. 389-406 1996.

[40] 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.

[41] Woszczynski, A. B., Guthrie, T. C., Chen, T. L. and Shade, S. Personality and Programming. Journal of Information Systems Education. Vol. 16 no. 3 pp. 293-299.

[42] Sodiya., A.S., Longe., H.O.D., Onashoga., S. A., Awodele. O. and Omotosho. L. O. An improved assessment of personality traits in software engineering. Interdisciplinary Journal of Information, Knowledge and Management. Vol. 2. pp. 163-177.

[43] Big Five personality theory. https://www.123test.com/big-five-personality-theory/ .

[44] The Myers & Briggs Foundation. http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/.

[45] Furnham, A., Jackson, C. J. and Miller, T. Personality, learning style and work performance. Journaly of Personality and Individual Differences. Vol. 27. pp. 1113 – 1122 1999.

Personality Questionnaire

1. I am seen as "outgoing" or as a "people person."

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

2. I feel comfortable in groups and like working in them.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

3. I have a wide range of friends and know lots of people.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

4. I sometimes jump too quickly into an activity and don't allow enough time to think it over.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

5. Before I start a project, I sometimes forget to stop and get clear on what I want to do and why.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

6. I remember events as snapshots of what actually happened.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

7. I solve problems by working through facts until I understand the problem.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

8. I am pragmatic and look to the "bottom line."

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

9. I start with facts and then form a big picture.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

10. I trust experience first and trust words and symbols less.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

11. Sometimes I pay so much attention to facts, either present or past, that I miss new possibilities.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

12. I enjoy technical and scientific fields where logic is important.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

13. I notice inconsistencies.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

14. I look for logical explanations or solutions to most everything.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

15. I make decisions with my head and want to be fair.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

16. I believe telling the truth is more important than being tactful.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

17. Sometimes I miss or don't value the "people" part of a situation.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

18. I can be seen as too task-oriented, uncaring, or indifferent.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

19. I like to have things decided.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

20. I appear to be task oriented.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

21. I like to make lists of things to do.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

22. I like to get my work done before playing.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

23. I plan work to avoid rushing just before a deadline.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

24. Sometimes I focus so much on the goal that I miss new information.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

25. I am seen as "reflective" or "reserved."

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

26. I feel comfortable being alone and like things I can do on my own.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

27. I prefer to know just a few people well.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

28. I sometimes spend too much time reflecting and don't move into action quickly enough.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

29.I sometimes forget to check with the outside world to see if my ideas really fit the experience.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

30. I remember events by what I read "between the lines" about their meaning.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

31. I solve problems by leaping between different ideas and possibilities.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

32. I am interested in doing things that are new and different.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

33. I like to see the big picture, then to find out the facts.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

34. I trust impressions, symbols, and metaphors more than what I actually experienced

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

35. Sometimes I think so much about new possibilities that I never look at how to make them a reality.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

36. I have a people or communications orientation.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

37. I am concerned with harmony and nervous when it is missing.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

38. I look for what is important to others and express concern for others.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

39. I make decisions with my heart and want to be compassionate.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

40. I believe being tactful is more important than telling the "cold" truth.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

41. Sometimes I miss seeing or communicating the "hard truth" of situations.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

42. I am sometimes experienced by others as too idealistic, mushy, or indirect.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

43. I like to stay open to respond to whatever happens.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

44. I appear to be loose and casual. I like to keep plans to a minimum.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

45. I like to approach work as play or mix work and play.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

46. I work in bursts of energy.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

47. I am stimulated by an approaching deadline.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

48. Sometimes I stay open to new information so long I miss making decisions when they are needed.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

Programming Attitude Questionnaire

1. Generally I have felt secure about attempting computer programming problems.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

2. I am sure I could do advanced work in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

3. I am sure that I can learn programming.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

4. I think I could handle more difficult programming problems.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

5. I can get good grades in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

6. I have a lot of self-confidence when it comes to programming.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

7. I'm no good at programming.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

8. I don't think I could do advanced computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

9. I'm not the type to do well in computer programming.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

10. For some reason even though I work hard at it, programming seems unusually hard for me.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

11. Most subjects I can handle O.K., but I have a knack for flubbing up programming problems.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

12. Computer science has been my worst subject.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

13. It would make me happy to be recognized as an excellent student in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

14. I'd be proud to be the outstanding student in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

15. I'd be happy to get top grades in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

16. It would be really great to win a prize in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

17. Being first in a programming competition would make me pleased.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

18. Being regarded as smart in computer science would be a great thing.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

19. Winning a prize in computer science would make me feel unpleasantly conspicuous.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

20. People would think I was some kind of a nerd if I got A's in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

21. If I had good grades in computer science, I would try to hide it.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

22. If I got the highest grade in computer science I'd prefer no one knew.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

23. It would make people like me less if I were a really good computer science student.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

24. I don't like people to think I'm smart in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

25. Females are as good as males at programming.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

26. Studying computer science is just as appropriate for women as for men.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

27. I would trust a woman just as much as I would trust a man to figure out important programming problems.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

28. Women certainly are logical enough to do well in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

29. It's hard to believe a female could be a genius in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

30. It makes sense that there are more men than women in computer science.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

31. I would have more faith in the answer for a programming problem solved by a man than a woman.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

32. Women who enjoy studying computer science are a bit peculiar.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

33. I'll need programming for my future work.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

34. I study programming because I know how useful it is.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

35. Knowing programming will help me earn a living.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

36. Computer science is a worthwhile and necessary subject.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

37. I'll need a firm mastery of programming for my future work.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

38. I will use programming in many ways throughout my life.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

39. Programming is of no relevance to my life.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

40. Programming will not be important to me in my life's work.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

41. I see computer science as a subject I will rarely use in my daily life.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

42. Taking computer science courses is a waste of time.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

43. In terms of my adult life it is not important for me to do well in computer science in college.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

44. I expect to have little use for programming when I get out of school.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

45. I like writing computer programs.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

46. Programming is enjoyable and stimulating to me.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

47. When a programming problem arises that I can't immediately solve, I stick with it until I have the solution.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

48. Once I start trying to work on a program, I find it hard to stop.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

49. When a question is left unanswered in computer science class, I continue to think about it afterward.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

50. I am challenged by programming problems I can't understand immediately.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

51. Figuring out programming problems does not appeal to me.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

52. The challenge of programming problems does not appeal to me.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

53. Programming boring.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

54. I don't understand how some people can spend so such time on writing programs and seem to enjoy it.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

55. I would rather have someone give me the solution to a difficult programming problem than to have to work it out for myself.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

56. I do as little work in computer science courses as possible.

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

Programming Style Questionnaire

1. What is your age? _____

2. What is your gender? _____

3. How many months/years have you been programming? ____year____months

4. How many programming languages have you ever contributed code larger than 100 lines of code?

☐ C/C++ ☐ Java ☐ C# ☐ Python ☐ CUDA ☐ OpenACC ☐ OpenMP ☐ MPI ☐ OpenCL ☐ others

If others, please list: _____

5. What is the largest numbers of lines of code you have made?

☐ <100 ☐100-1000 ☐1000-5000 ☐>5000

6. How do you roughly estimate your grade of all exams? General class performance in programming

☐ average ☐ above average

7. I have developed software on my own:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

8. I have developed software as part of a team and/or as part of a course:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

9. I prefer to program with a group:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

10. I prefer to program alone:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

11. Facing bugs, I usually think alone:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

12. Facing bugs, I usually search them online:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

13. Facing bugs, I usually ask friends/instructors:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

14. I usually program continuously within hours:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

15. I usually program intermittently

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

Imagine you will publish a big project (Q16 – Q17).

16. You prefer to maintain it in the way of open source i.e. Ubuntu

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

17. You prefer to maintain it in the way of closed-source i.e. Windows

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

18. What programming context do you like? Visual i.e. Visual Basic

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

19. What programming context do you like? Text-based i.e. vi/vim in Ubuntu

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

20. For testing code, I prefer to test it unit by unit:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

21. For testing code, I prefer to test it by all units at a time:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

22. Imagine you are working on projects with a group:

a. I like to program with members together in an office:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

b. I like to program with members through internet:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

c. To explore requirements, I like to draw sketches on whiteboards:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

d. To explore requirements, I like to have a talk with members

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

e. While programming, I like to write efficient code but hard for members to understand:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

f. While programming, I like to write more code so that members easily understand

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

g. I like to synchronize my code with other members with an automatic tool i.e. Github with extra coding workload:

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree

h. I like to synchronize my code with USB drives or email (no extra coding workload):

A. Disagree B. Little Disagree C. Neutral D. Little Agree E. Agree