CRITICAL PERSONALITY TRAITS IN SUCCESSFUL PAIR PROGRAMMING

Gulgunes Atli

A Thesis

Submitted to the Graduate College of Bowling Green
State University in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

August 2006

Committee:

Joseph Chao, Advisor

Jane Chang

Laura Leventhal

© 2006

Gulgunes Atli

All Rights Reserved

ABSTRACT

Joseph Chao, Advisor

In Pair Programming (PP) two programmers are working together side by side on the same task using a single computer. PP is one of the twelve practices in Extreme Programming which promises to produce reliable, efficient, and well-factored software products. The benefits of PP are well documented in experimental studies. Some of the benefits include improving design quality, reducing defects, reducing staffing risks, enhancing technical skills, and improving team communications. Pair programming has become increasingly popular in the field; however, some programmers are skeptical of pairing. Many believe that pair programming success highly depends on the compatibility of the pairs, which is determined by the personality traits of the partners.

In this research, personality traits' effects on pair programming success were studied empirically. The hypothesis is that certain personality traits are crucial for the success of PP, and PP partners should be chosen based on these personality traits. Programmers in the software development industry were surveyed to identify the perceived important personality traits for successful PP. A series of experiments were conducted among undergraduate students in computer science classes to determine the significance of these personality traits in successful PP. The successfulness in PP has been evaluated by measuring the quality of the produced code and the pairs' compatibility. Then, the collected data were analyzed through ANOVA and multiple comparison tests. The study found that there are some indications that personality traits

of the pairs influence success in pair programming in terms of producing high quality codes and pairs' compatibility.

To my family and Daniel

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Joseph Chao, for his support, helpful suggestions, and his patience with my writing. Also, I wish to acknowledge Dr. Jane Chang for her great help in data analysis part of the study. I would like to thank Dr. Laura Leventhal for her helpful criticism and comments.

TABLE OF CONTENTS

	Page
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. LITERATURE REVIEW	5
CHAPTER 3. EXPERIMENTS	8
3.1 Identification of the perceived five most important personality traits	9
3.1.1 On-line Surveys	9
3.1.2 Survey Results.....	11
3.1.3 Perceived Important Personality Traits.....	12
3.2 Recruit Process of the Participants.....	13
3.2.1 Personality Traits of Participants	13
3.2.2 Correlation among Personality Traits	14
3.3 Experimental Design.....	16
3.3.1 The Four Experiments.....	17
3.3.2 Quality of Code and Pair Compatibility	19
CHAPTER 4. STATISTICAL ANALYSIS	21
4.1 One-Way ANOVA.....	22
4.1.1 Personality Traits versus Quality of Code	22
4.1.2 Personality Traits versus Pair Compatibility	23
4.2 Pairwise Multiple Comparison	24
CHAPTER 5. CONCLUSIONS	25

CHAPTER 6. RECOMMENDATIONS FOR FUTURE STUDIES	27
REFERENCES	29
APPENDIX A. ONLINE SURVEY TO IDENTIFY THE PERCEIVED FIVE MOST IMPORTANT PERSONALITY TRAITS IN PP	31
APPENDIX B. STUDENT PARTICIPANT CONSENT FORM	33
APPENDIX C. PAIR PROGRAMMING HANDOUT	34
APPENDIX D. PERSONALITY TRAITS TEST	37
APPENDIX E. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 1	39
APPENDIX F. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 2	40
APPENDIX G. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 3	41
APPENDIX H. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 4	43
APPENDIX I. STATISTICAL ANALYSIS TABLES	45

LIST OF TABLES

Table	Page
Table 1: Fifteen Personality Traits.....	10
Table 2: Perceived Five Most Important Personality Traits	12
Table 3: Personality Scores.....	14
Table 4: Correlation Coefficient Values	15
Table 5: Final four personality traits.....	16

LIST OF FIGURES

Figure	Page
Figure 1: Experiment Flowchart.....	8
Figure 2: Survey participants' PP experience.....	11
Figure 3: Creative and Open-minded linear relation.....	15

CHAPTER 1. INTRODUCTION

In Pair Programming (PP), two programmers work collaboratively at one computer on the same design, algorithm, or code. While one member of the pair, referred to as the driver, types the code, the other member, the navigator, reviews the code against syntax/logical/design errors, asks questions, raises objections, and suggests alternative ways. More precisely, the navigator thinks about broader issues, i.e., how the current code will fit the existence code, how the new lines of code will be tested, what should be done after this part of code has been written, and will the existing code need to be refactored to adapt the new code. The driver and navigator are in continuous communication during the pairing process: asking and answering questions about the design or coding problems. Pair programmers can change their roles as driver or navigator during pairing. If the navigator sees that the driver gets stuck, then this might be an indication that it is time to change the roles. Changing roles is a crucial part of the methodology; pairs are required to spend roughly equal amounts of time driving and navigating. In addition, the combination of partners can change depending on the certain tasks and situations like the illness of one partner.

Pair programming does not mean dividing the task in half and each programmer is doing one part. In pair programming, every task should be done together by two programmers. This approach has many benefits to programmers. Pair programmers always have someone with whom to discuss the problems. Pair programming practice allows programmers to discuss and come up with better code and design. The act of expressing their ideas aloud to their partners helps them clarify their ideas and leads them to see pitfalls.

Pair programming is a practice which was introduced by Agile Software Development, a relatively new software development methodology. It presents a new way of developing

software to traditional software development methodologies. Traditional methods focus on planning the future in great detail. Even a little change in the requirements and direction can cause major problems and eventually the failure of a project. Agile developers believe that frequently changing requirements is the nature of the software development process. Therefore, the development method should be able to adapt changes into development cycle without causing major problems. Contrary to traditional predicate methods, agile methods can adapt to changes quickly.

Agile developers place more value on individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [1]. Principles of agile include customer satisfaction, welcoming changing requirements, delivering working software frequently, face-to-face conversation, good design, and simplicity. Agile methods are “intended to support early and quick production of working code by structuring the development into small release cycles and focus on continual interaction between developers and customers [2].” Agile methods deliver a working software product in short time boxes, which are called iterations. Depending on the agile methods, duration of time boxes varies, but most typically a single iteration takes one to four weeks. All the steps of the classical life cycle model: requirement analysis, design, code generation, testing and maintenance, are followed in each iteration. The project team reevaluates the priorities and requirements of the project at the end of iterations.

Extreme Programming (XP) is probably the most popular agile software development methodology. XP has 12 major practices, and these practices aim to promote communication, simplicity, feedback, courage, and respect which are the five values of XP. PP is one of the XP practices, which benefits the XP team in several ways: increasing the discipline of team

members, mentoring, team cohesion, and a more enjoyable environment. Communication between developers, quick feedback, and collective code ownership in XP are mainly accomplished by PP practice rather than through documentation. XP techniques aim to give all developers and the users a view of the system by encouraging them to communicate face to face. XP encourages programmers to code for today's requirements of the system. In other words, since requirements frequently change during the process, placing less time and effort at the beginning is recommended. In short, the methodology encourages starting with the simplest possible code and improving it throughout the process according to new requirements. The encouragement of coding for today's requirements increases confidence of programmers with their code. When the programmers are confident, they feel comfortable refactoring the code. In XP, all the programmers are the owners of the code, and this practice is called collective code ownership. Programmers in XP team show respect to each other because of the collective code ownership idea.

While PP has become increasingly popular in the field, developers and project managers have questioned the elements that predict a pair's compatibility. Programming skill levels and personality traits of the partners are two important elements that can affect success in PP. To ensure the success, the critical personality traits in successful PP should be known.

In this research we study the impact of personality traits on pair programming, more precisely, the impact on the quality of the code produced by the pair, and on the satisfaction of the pair with the experience of pair programming with a particular partner. The hypothesis is that pairing programmers based on certain personality traits helps ensure the success of pair programming. The study result will help the software development industry to match the pair

programming partners in a way which improves the chances of getting high quality software products and having compatible pairs.

We asked experienced pair programmers in the software development industry and undergraduate students with one semester of pair programming experience to determine the set of personality traits that seemed critical for successful pair programming. Sixty-five programmers in the industry and twenty-two undergraduate students chose the five most important personality traits in PP. Then we carefully selected the five personality traits that had frequently appeared in both lists of the two groups. Four experiments were conducted to determine the significance of the most important traits that were selected by the programmers in the industry and undergraduate students with pair programming experience.

In Chapter 2, we summarize previous work on the topic of personality traits and pair programming relation. Also, the value of our study within the field of software development is explained. Chapter 3 presents the steps that we followed in the experiments. Data analysis was presented in Chapter 4. Chapter 5 summarizes the study and discusses recommendations for future studies.

CHAPTER 2. LITERATURE REVIEW

PP improves the quality of the produced code, knowledge transfer between programmers, and communication throughout the whole team. Benefits of PP versus solo programming are demonstrated in previous studies. Williams et al. [3] showed that pair programmers produce higher quality software products than solo programmers in a shorter amount of time. Cockburn et al. [4] studied the costs and benefits of pair programming. They found that PP improves design quality, reduces defects, reduces staffing risk, enhances technical skills, and improves team communication for a very little development-time cost. They also observed that programmers enjoyed software development more when they pair programming. In addition to providing high quality codes, PP encourages team members to communicate and transfer their knowledge to each other to increase the effectiveness of the team.

Gallis et al. [5] studied the factors impacting team programming success, and presented a framework including major factors such as education, experience, and communications. They also suggested that team members' personality impacts on team success should be investigated. Ally et al. [6] extended the initial framework and treated the suggested factors from a pair programming point of view. They conducted a Delphi study of pair programming among 20 participants. The Delphi is an evaluation technique and allows the researcher to reach a trustworthy general agreement from a panel of experts. It has been well-documented in [7] and [8]. The findings of Delphi pair programming study supported the idea of investigating personality impacts on pair programming success. Moreover, in the extended framework, personality factor was branched into two separate factors as pair personality and resistance to pair programming. They also suggested that the effects of the pairs' personality in pair

programming success should be investigated. They investigated two specific issues in personality aspects of PP: personality conflicts and divergent personality styles. Personality conflict issue has been explained as “when two people have different ideas, or different styles of dealing with problems” and “some people just don’t like accepting other people’s suggestions or ideas.” In divergent personality styles, “some programmers like to discuss with others, while some do like to work on issues by themselves”.

To increase the chance of forming compatible pairs in classrooms, Katira et al. [9] studied the relationship between pair programming and personality traits to understand if pairs are more compatible when the pairs are grouped together with different personality types. Over 564 undergraduate and graduate students in computer science classes participated in the study. Myers Briggs Type Indicator (MBTI) was used to classify students into four dimensions of personality types: Introvert-Extrovert, Sensing-Intuition, Thinking-Feeling, and Judging-Perceiving. According to their result in Introduction to Programming-Java (CS1) class, pairs were more compatible if pairs’ had different personality types ($p < 0.039$). However, the same result was not found for the pairs in undergraduate software engineering class. Katira et al. concluded that pair compatibility in introductory courses may increase if pairs have dissimilar personality traits.

Williams et al. [10] presented the second phase of Katira et al.’s study. Researchers continued to investigate their hypothesis, which is pairs are more compatible if the pairs with different personality types are grouped together, with more data samples. Their hypothesis was that different personality types should be grouped together so that they complement each other and succeed in producing better results. The overall result showed that Myers Briggs’s four dimensions personality types do not indicate any relation between personality types of the pairs

and compatibility. However, a significant correlation was found between a pair's sensing-intuition level and whether they were very compatible or not.

A recent study on personality traits and pair programming in the industry has been done by Dick and Zarnett [11]. According to their study not all programmers are suited for pair programming and team members should be selected based on personality traits. They argued that successful pairing is difficult to achieve, and programmers for pair programming should be selected with personality traits that are beneficial to pair programming. According to their qualitative observations, effective communication, comfort in working with others, self confidence, and ability to compromise are the four essential traits for effective pair programming.

Although there have been some investigation of personality effects on pair programming success, more are needed in the field. The previous studies investigate the personality traits and pair programming relation in terms of pairs' compatibility. In previous studies such as [9] and [10], only the compatibility of the pairs was investigated as an indicator of success in pair programming. In our research, besides pair's compatibility, the quality of the produced code was also measured to determine the relationship between personality traits and successfulness in PP. Previous experimental studies in the field determined the experiment participants' personality according to Myers Brigg's personality types. In our study, we emphasized on individual personality traits.

CHAPTER 3. EXPERIMENTS

For this research, we first identified the perceived five most important personality traits by conducting an online survey of programmers in industry. Then we recruited students to participate in the study. We determined students' personality traits level by conducting online personality traits test. We investigated possible correlation among personality traits. After determining personality traits level of the students, we matched them on their personality traits levels. We assigned pair programming assignments for each pair to complete. After the completion of the assignments, we asked pairs to evaluate their compatibility. We also prepared guidelines to evaluate their solution to assignments. Figure 1 summarizes the steps.

Figure 1: Experiment Flowchart

3.1 Identification of the perceived five most important personality traits

To identify the perceived five most important personality traits in pair programming, an online survey was conducted among professional programmers in software industry.

3.1.1 On-line Surveys

To narrow down the choices for the online survey, we selected fifteen personality traits that are relevant to PP success from the list of thirty-nine personality characteristics prepared by University of Denver Career Center [12]. The survey participants were provided with the list of fifteen personality traits, accompanied by descriptions as shown in Table 1. Invitations to the survey were sent to Michigan eXtreme Programming Enthusiasts (MXPE) and Agile Modeling mailing groups. Major part of the survey participants were the members of these mailing groups. The participants were asked to select and rank “the five most important traits that you would like your pair programming partner to have.” A complete copy of the online survey is given in Appendix A.

Personality Trait	Description
ACTIVE	Enthusiastic, dynamic, lively, agile, high energy, busy.
ADVENTUROUS	Try new things, take risks.
ATTENTIVE	Listen well, pay close attention to verbal and non-verbal messages.
CALM	Remain peaceful, composed, and serene under pressure.
COMPETITIVE	Strive to win, oppose a rival.
CREATIVE	Imaginative, inventive, original.
DECISIVE	Show firm determination in using judgment to make decisions.
DIPLOMATIC	Good communication skills, tactful, skillful at negotiating.
EMPATHIC	Compassionate, sensitive to or can identify with the feelings of others.
FLEXIBLE	Adaptable, versatile.
LOGICAL	Rational, analytical, reasonable.
OPEN-MINDED	Impartial, receptive to new ideas, free from prejudice or bigotry.
ORGANIZED	Structured, orderly, systematic.
OUTGOING	Sociable, friendly.
RESPONSIBLE	Dependable, trustworthy, reliable, loyal.

Table 1: Fifteen Personality Traits

We wanted to know participants' PP experience level therefore we asked how long participants had experienced in PP. According to survey results, 22% of the participants have 3 or more years of experience, 20.3% have 1 or more years of experience, 8.5% 6 or more months of experience, 13.6% less than 6 months of experience, and 35.6% do not have any pair programming experience. Figure 2 shows the number of participants in percentage and the years of experience they have in pp.

Figure 2: Survey participants' PP experience

3.1.2 Survey Results

Based on responses from 60 programmers in the software industry who participated in the survey, we found that 75% of the participants ranked Open-minded as the most important personality trait followed by Creative, Attentive, Logical, and Flexible. However, if we separate the programmers based on their experience with pair programming, the results were slightly different. Although both experienced and non-experienced groups rank Open-minded and Creative as their top two choices, the experienced group placed a higher value on Attentive, Logical, and Active, while the non-experienced group ranked Logical, Responsible, and Attentive as the other three most important personality traits. Note that “non-experienced” refers to that those respondents who had little experience in pair programming.

Because subsequent phases of the study were planned around students as the subjects for the personality trait experiments, the same online survey was administered to students in a programming course with one semester of pair programming experience. Twenty-two students

took the survey and ranked the five most important traits in the order of Responsible, Logical, Open-minded, Flexible, and Attentive.

A summary of the survey results are given in Table 2.

Rank	Overall Industry	Experienced Industry	Non-experienced Industry	Students
1	Open-minded	Open-minded	Open-minded	Responsible
2	Creative	Creative	Creative	Logical
3	Attentive	Attentive	Logical	Open-minded
4	Logical	Logical	Responsible	Flexible
5	Flexible	Active	Attentive	Attentive

Table 2: Perceived Five Most Important Personality Traits

3.1.3 Perceived Important Personality Traits

The findings were somewhat consistent in the sense that Open-minded, Logical, and Attentive were on all lists, and Creative, Responsible, and Flexible were on at least two of the lists.

Notice that Responsible was the top choice of students and the fourth choice of the non-experienced pair programmers in industry, but Responsible was not among the top five traits selected by experienced pair programmers in industry. One possible explanation is that there are less serious ramifications for lack of responsibility in an academic environment, meaning that there are a higher percentage of irresponsible students than professionals. Creative was highly ranked by industry programmers but not by students. This difference might be attributed to the fact that student assignments in introductory programming classes are typically not very complicated, and therefore creativity was not perceived as being as important.

The final list of personality traits selected for this study consists of Open-minded, Creative, Logical, Responsible, and Attentive. Open-minded, Logical, and Attentive appear on all survey groups' top five personality traits, and Responsible and Creative were included in light of the

preceding discussion. Although Flexible appeared on more than one list, it was not deemed to be highly rated enough to be included in the study.

3.2 Recruit Process of the Participants

The experimental design for this study was submitted to and approved by the Human Subjects Review Board at the university level in spring 2006. Upon receiving approval, we recruited 70 students in CS1 and 34 students in CS2. The consent form is available in Appendix B. In total, 104 undergraduate students were recruited. However, the number of participants was reduced to 88 (44 pairs) due to students' dropping classes, among other reasons. Four experiments were conducted in this study. CS1 students participated in Experiments 1 and 2; CS2 students participated in Experiments 3 and 4. The concept of pair programming was explained to the students in detailed prior to the study. The handout provided to students is available in Appendix C.

3.2.1 Personality Traits of Participants

First, the personality traits of the participants were determined so that they could be paired accordingly. A personality trait test was prepared and posted online for the participants to take. The online personality traits test is given in Appendix D. The test contained a total of 40 statements and we asked participants to evaluate how accurately each statement describes them. The statements were carefully selected from a website maintained by the International Personality Item Pool [13].

For each five traits under study, we provided eight statements. Among these eight statements, four represented positive aspects of the personality traits, the other four represented negative aspects of the personality traits. Participants were provided 5 answer choices, ranging from “moderately inaccurate” to “moderately accurate”. The responses for positive statements were scored from 1 “moderately inaccurate” to 5 “moderately accurate”. For negative statements the point scale was used in reverse order as 1 to “moderately accurate” and 5 to “moderately inaccurate”. After all test results were tallied, each participant was assigned five scores, one for each of the personality traits. In Table 3, we demonstrated personality traits levels of students X and Y. If student’s personality trait level is lower than the determined average point, then we assigned low (L), if it is higher then we assigned high (H).

Student	Open-minded	Creative	Logical	Responsible	Attentive	Open-minded	Creative	Logical	Responsible	Attentive
X	75	75	63	74	90	L	L	L	H	H
Y	60	85	70	50	80	L	H	H	L	H

Table 3: Personality Scores

3.2.2 Correlation among Personality Traits

One personality trait could be highly related to another; therefore, the existence of correlation among personality traits was investigated to prevent bias in the experiment. A stepwise regression analysis was performed to investigate the possible existence of correlation between traits using the personality trait test result. The resulting correlation coefficients between each pair of the five personality traits are given in Table 4.

	Open-minded	Creative	Logical	Responsible
Creative	0.709			
Logical	-0.135	-0.090		
Responsible	0.325	0.204	-0.098	
Attentive	0.321	0.277	-0.070	0.322

Table 4: Correlation Coefficient Values

As shown in Table 4, the correlation coefficient between Open-minded and Creative is significantly high. The significant correlation means that we can predict Open-minded from Creative or vice versa. The non-significant correlations mean that we can not predict one factor from another. Figure 2 shows strong positive linear relation between Creative and Open-minded.

Figure 3: Creative and Open-minded linear relation

In other words, the data showed that participants who have a high rating for the Open-minded trait are likely to have a high score for the Creative trait as well. As a result, the Creative trait was integrated into the Open-minded trait to prevent dependency in the variables. The

combination of the two traits resulted in the following four personality traits is as shown in Table 5.

Final Four Personality Traits
Open-minded
Logical
Responsible
Attentive

Table 5: Final four personality traits

3.3 Experimental Design

For each personality trait in the study, participants were categorized into two levels: high or low. For instance, a participant might be high Open-minded, low Attentive, low Responsible, and high Logical. For each personality trait, there were three possible pair combinations: High/High, High/Low and Low/Low.

Participants were paired based on their personality traits according to the following experimental design in all four experiments. We attempted to have as equal as possible the numbers of pair combinations (High/High, High/Low and Low/Low as discussed above) in each personality trait to maximize the amount of information that could be obtained statistically. The participants were matched with different partners in each experiment. In short, we had different pairs in four experiments. The idea was to minimize the influence of the special pair situations to the overall data result.

This study employs one independent and two dependent variables. The independent variable in this study was the combination of personality attributes in group. Independent variable had 3

different values: LL (both partners have a low level of the trait), HL (one pair partner has a high level of the trait and the other has a low level of the same trait), and HH (both partners have a high level of the trait). Dependent variables are quality of the code produced by the pairs and pairs' compatibility. Because each pair were experienced one level of the independent variable, the experimental design was "between subjects". In other words, one pair combination can be either HL, or HH, or LL in one particular personality trait.

3.3.1 The Four Experiments

We conducted four experiments in the study. Experiments 1 and 2 were among CS1, Experiments 3 and 4 were among CS2 undergraduate students. The idea of conducting experiments in two different level classes was to investigate whether there is a relation between the complexity level of classes and personality traits' effects on success in pair programming.

In each experiment, pair programming assignments were given to participants. These assignments were part of regular class lecture flow and were created by the instructors. The assignments were designed to test students' acquisition on new class material. Because the first assignments were considered easy because of relatively easy class material at the beginning of the course, we ran the experiments in the third or above assignments. The idea was that when the assignment is harder, the evaluation of the solutions is more deterministic to distinguish pairs' work.

Experiment 1 was started at the end of the sixth week of the semester. Participants were 58 undergraduate students in three sections of CS1. The two sections of these classes had been taught by the same instructor. The schedule was carefully designed to allow students to become comfortable with pair programming through weekly paired programming labs before the

experiment. After pairing the students according to their personality traits test levels, pair programming assignments were given to the pairs. The assignment was the students' third assignment and students were asked to develop a graphical clock by using Point, Line, Circle, Message, and GraphicWindow classes in C++. The objective of the assignment was to teach usage of objects and their member functions. A one-week time period was given to the pairs to finish the assignment.

Fifty-two undergraduate students (26 pairs) in three sections of CS1 participated in Experiment 2. However, six pairs were eliminated from data analysis at the end because they could not do pair programming more than one hour. Because Experiments 1 and 2 were among the same group of students, pairing process has been done carefully to have different pairs than Experiment 1. The pair programming assignment was to develop the children's game Chutes and Ladders in C++ and it was students' fifth assignment in the course. The objective of the assignment was to teach functions, stepwise refinement, passing parameters to functions, function calls, scope of parameters, and the use of objects. A one-week time period was given to the pairs to finish the assignment.

Thirty-six undergraduate students (18 pairs) in two sections of CS2 participated in Experiment 3. Three pairs' data were discarded before the data analysis because they did pair programming less than one hour. The assignment was students' third assignment and pairs were asked to create a vector of student objects and process the vector based on interactive menu functions such as add a student to list, sorting student list by name, listing all students, listing students by a given classification, list Dean's List students, and printing students above the average GPA. A two-week time period was given to the pairs to complete the assignment.

Twenty-eight undergraduate students (14 pairs) in two sections of CS2 participated in Experiment 4. The same instructor taught both sections. Because the same students participated in Experiments 3 and 4, the pairs were matched differently in two experiments. Three pairs out of 14 did pair programming less than 1 hour; therefore, their data were eliminated at the end. The assignment that was given to the pairs was students' fifth assignment, and asked pairs to work on user-defined linked list by implementing `push_front`, `pop_back`, and `get_size` functions.

3.3.2 Quality of Code and Pair Compatibility

Two major criteria were used to measure the success of pair programming: the quality of the code produced and the compatibility of the pairs.

In order to measure the quality of the code, guidelines were prepared and made available to students so that they were aware of how the quality would be measured. Major areas of focus included output correctness, required documentation, good programming style, correct use of objects, and interface design. The guidelines for four experiments are presented in Appendix E. The measuring was done by a single person to ensure consistency for each experiment. Scores for code quality varied for each experiment because every assignment had its own requirements and level of hardness. The max possible score for the code quality in the Experiment 1, Experiment 2, Experiment 3 and Experiment 4 were 75, 100, 125, and 102 respectively.

The compatibility measurement was done by an online survey administered to participants immediately after the assignments were turned in. Questions such as "I enjoyed pair programming with my partner" and "My partner and I were a perfect match" were included in the survey. Each compatibility assessment was based on a Likert scale. In the compatibility survey, we also asked how many hours they actually did pair programming. If the pairs

conducted pair programming for less than one hour, their data was eliminated because it is very unlikely that they accomplished all the rules in pair programming such as debating, discussing the problems, and switching the roles in less than one hour. In Experiment 1, the Likert scale ranged from 1 to 4 and in the other three experiments, the range was broadened from 1 to 10.

CHAPTER 4. STATISTICAL ANALYSIS

Since in each experiment the students were from different classes, which could result in confounding effects among the different classes (blocks), an Analysis of Variance (ANOVA) test was conducted to detect a potential “blocking effect.” The test revealed that no significant relation between classes and quality scores exists. Thus, for each experiment, the data from different classes was analyzed altogether.

Both the quality-of-code scores and the compatibility scores were analyzed with respect to each of the four personality traits. Recall that for each personality trait there were three possible pairing combinations: LL (both partners have a low level of the trait), HL (one partner has a high level, the other one has a low level of the trait), and HH (both partners have a high level of the trait). The objective was to determine for each personality trait whether the pair programming results would be different if pairs are matched according to their level of that trait.

The null hypothesis can be stated as:

$$H_0: LL = HL = HH$$

The objectives are to see whether there are significant differences for the three ways of pairing particular personality traits, first, in the quality of code produced, and second, in the pairs' compatibility. In addition, pairwise multiple comparison tests for the three groups were conducted, based on equal and unequal variance assumptions.

4.1 One-Way ANOVA

For each experiment's dataset, four ANOVA tests based on the personality trait level were performed twice, first, against the code quality score, and second, against the pair compatibility score. The code quality scores and pair compatibility scores, respectively, are treated as the response variables, whereas the personality traits are treated as factors having three different treatments: LL, HL, and HH. Since data shows potential non-consistent variances between the treatments and one-way ANOVA assumes equal variance between the treatments, tests using Proc Mixed of the SAS System were also conducted to consider unequal variances between the three different personality trait pairings. The P-values differs between two tests' results. Since data shows some differences among variances between treatments, it has been decided to consider only the P-values that were generated under unequal variance assumption.

In order to be able to assume that personality traits influence the success of pair programming, the null hypotheses (H_0) must be rejected. If the P-value (observed significant value) is less than or equal to the chosen significance level α , the null hypotheses can successfully be rejected.

4.1.1 Personality Traits versus Quality of Code

The ANOVA results for quality of code show that H_0 cannot be rejected ($\alpha = 0.05$) in any of the four experiments. However, in Experiment 4 for the Open-minded personality trait, the P-value is very close to the significance level 0.05 (P-value = 0.0559 under unequal variance). Although it is not significant, pairs that had two partners low in Open-minded personality traits

produced less quality code than the pairs that had both partners high in Open-minded or one partner low and the other one high in Open-minded.

There are other indications that the average mean score in quality of code varies for different pairing groups. For example, in Experiment 1 for the Open-minded personality trait, the average quality of code scores for HL and HH pairs is higher than for LL pairs; the average scores of HH and HL are 68.000 and 68.100, respectively, compared to the 65.455 of LL. In Experiment 2, the average score of HL pairs' code quality is substantially higher than that of LL and HH pairs in the Logical personality trait; the average scores of HH and LL pairs are 86.50 and 88.00, respectively, compared to the 93.33 of HL. Detailed data is presented in Appendix I.

4.1.2 Personality Traits versus Pair Compatibility

The ANOVA results for the pair compatibility score versus each of the four personality trait groups show that H_0 cannot be rejected ($\alpha = 0.05$) in any of the four experiments. However, in Experiment 2 for Open-minded personality trait, observed P-value is close to the chosen significance level α (P-value = 0.0639 under unequal variance). Although it is not significant, according to the Experiment 2 dataset results for Open-minded personality trait, pairs with mixed levels of personality trait and low levels of personality trait were less compatible than the pairs with both partners high in Open-mindedness. Detailed data is presented in Appendix I.

4.2 Pairwise Multiple Comparison

Multiple pairwise comparison tests under equal and unequal variance assumptions were conducted for the datasets of all four experiments under equal and unequal variance assumptions, in order to determine the differences between the three groups of personality traits. In Experiment 4, for code quality, H_0 was rejected ($\alpha = 0.05$) for HL versus HH Open-minded pairs (P-value = 0.0478 under unequal variance), i.e., there is significant evidence that pairs with two high levels in Open-minded produce higher quality code than pairs with mixed levels. Although it is not significant, in Experiment 2 for Open-minded personality trait, the P-value for pairs, compatibility is close to the chosen significance level α (P-value = 0.0518 under unequal variance). In other words, although it is not significant, pairs that had both partners high in Open-minded were more compatible than the pairs that had both partners low in Open-minded. See Appendix I, Tables 9 to 16, for the P-values from pairwise comparison tests under equal and unequal variance assumptions.

CHAPTER 5. CONCLUSIONS

In this research, we conducted an empirical study on personality traits' effects on pair programming success. In this study we first identified the perceived five most important personality traits for pair programming by surveying both programmers in the industry as well as university students in programming classes. According to the survey results, we decided to focus on four personality traits in the study: Open-minded, Logical, Responsible, and Attentive. Four experiments were conducted in two introductory computer science courses. First we determined participants' personality levels in the four personality traits. Then participants were paired according to their personality traits level. We assigned pair programming assignments to pairs. To determine the success of pair programming, we measured both the quality of the code produced and the compatibility of the pairs. Statistical analyses were then performed to determine whether or not matching pairs according to their personality traits affected the success in pair programming.

This study has one significant result: in Experiment 4, significant evidence has been found that pairs with both partners having high levels of Open-mindedness produce code of higher quality than pairs with mixed levels of Open-mindedness. Although those findings are not consistent among all four experiments, they indicate that: pairs with high Open-mindedness will produce better code than pairs with mixed or low levels of Open-mindedness.

There are several possible explanations for the failure of rejecting the null hypothesis.

First, consider the type of the tasks. We conducted the experiments in introductory computer science classes, and the goal of the assignments was to teach programming, not to professionally produce software. Thus, the assignments were set up differently than in the industry. They were

designed for testing the students' knowledge, as well as for teaching. The instructors usually provided a very detailed description of the task. They even provided a preferred design. This situation differs clearly from the real software development cycle. Analysis and design are important parts of software production in the industry. Ideally, assignments in the experiments should be similar to software production procedure in industry.

Second, the participants in the experiments are presenting a different group than programmers in the software development industry. In the experiments, participants were novice programmers. It has been observed that novice programmers have different experiences with the practice of PP. During an on-line survey, students and programmers from the industry have separately been asked about their opinion on the five most important personality traits in PP. The results of the two groups were different. This indicates that the experience of students with PP differs from that of professional software developers. It is likely that students use different, non-optimal pair programming strategies for their learning tasks.

A third explanation might be that for novice programmers the difference in skill level has a greater impact on the quality of code than the difference in personality traits. The experiments were conducted in introductory classes in which many students participated that were not majoring computer science. Therefore the technical skill levels of the students varied widely. To compensate for this potential concern in the future, experiments should be conducted on experienced programmers in the industry. Ideally, the participants in the experiments possess the same level of expertise so that the effects of different personality traits become more visible. Conducting an experiment among experienced programmers from the industry would equalize the effect of skill level on pair programming success and would improve the external validity of the study.

CHAPTER 6. RECOMMENDATIONS FOR FUTURE STUDIES

Future studies should consider conducting experiments among participants with similar skill levels, with a higher number of participants, and with assignments similar to those in the software industry. Sensitivity of the measurements in determining the personality trait levels of the participants and the methods for evaluating the quality of code and compatibility of the pairs could also be improved in future studies.

The recommendation to conduct experiments among participants who have similar programming skill levels is based on the insight from Chapter 5, that if the participants' skill levels are the same, then personality traits affect the success of PP more. Experiments in graduate level classes could validate this argument.

The suggestion to conduct experiments with a higher number of participants is to increase the precision of the results. A higher number of observations might decrease the variance of the responses and point estimators of the treatment effect. Under such circumstances, it might be more meaningful both, when the null hypothesis is rejected and when not.

As discussed earlier, the external validity might be increased with assignments that are similar to those in the industry. Since requirements analysis, design, and testing are important parts of professional software development, those aspects should be part of the assignment and explanations in greater detail should be avoided, so that the participants are required to use their own creativity to solve problems. When the assignments are complex enough, sensitivity of the code quality measurement can be increased by evaluating the assignment solutions in detail.

The final recommendation for future studies concerns the environment in which the experiments are conducted: closed lab assignments should be preferred to homework

assignments. Ideally, in closed lab assignments all the pairs work in the lab during a fixed time span. Such proceeding should help to increase the sensitivity of measurement of the pairs' compatibility score by allowing a close observation of the interaction of the partners, so that, in addition to the compatibility surveys, the pair compatibility can be evaluated by qualitative judgment.

REFERENCES

- [1] Agile Alliance, “Manifesto for Agile Software Development,” 2001 [Online]. Available: <http://agilemanifesto.org/>. [Accessed: May 2006].
- [2] P. Sadasivan, “Efficacy of Pair Programming in Completing Programming Tasks,” *Bulletin of Applied Computing and Information Technology*, Vol. 3, Issue 1. ISSN 1176-4120 [Online]. Available: www.naccq.ac.nz/bacit/0301/2005Sadasivan_PairEfficacy.htm. [Accessed: May 2006].
- [3] L. Williams, R. Kessler, W. Cunningham, R. Jeffries, “Strengthening the Case for Pair Programming,” *IEEE Software*, Vol. 17, pp.19-25, July/August 2000.
- [4] A. Cockburn and L. Williams, “The Cost and Benefits of Pair Programming,” in *Proceedings of XP2000*, Sardinia, Italy, June 21-23, 2000.
- [5] H. Gallis, E. Arisholm, and T. Dyba, “An Initial Framework for Research on Pair Programming,” in *Proceedings of the 2003 International Symposium on Empirical Software Engineering-ISESE*, pp. 132-142, 2003.
- [6] M. Ally, F. Darroch, and M. Toleman, “A Framework for Understanding the Factors Influencing Pair Programming Success,” in *Proceedings of the 6th International Conference, XP 2005*, Sheffield, UK, June 18-23, 2005.
- [7] L. Day, “Delphi Research in the Corporate Environment, the Delphi Method: Techniques and Applications,” Addison-Wesley, London, 1975.
- [8] G. Rowe, G. Wright, and F. Bolder, “Delphi: Reevaluation of Research and Theory, Technological Forecasting and Social Change,” 1991.

- [9] N. Katira, et al. "On Understanding Compatibility of Student Pair Programmers," in *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, March 2004.
- [10] L. Williams, L. Layman, J. Osborne, N. Katira, "Examining the Compatibility of Student Pair Programmers," in *the Agile 2006 International Conference*, 2006.
- [11] A. Dick, and B. Zarnett, "Paired Programming and Personality Traits," in *XP2002*, Sardinia, Italy, 2000.
- [12] University of Denver Career Center, "Personality Characteristics," [Online]. Available: www.du.edu/career/handouts/Personality.pdf. [Accessed: February 2006].
- [13] Oregon Research Institute, "International Personality Item Pool," [Online]. Available: www.ipip.ori.org [Accessed: May 2006].

APPENDIX A. ONLINE SURVEY TO IDENTIFY THE PERCEIVED FIVE MOST IMPORTANT PERSONALITY TRAITS IN PP

The following is the online survey for identifying the perceived five most important personality traits.

Personality Traits in Pair Programming

By answering six simple questions in this survey, you will help us to determine the five most important personality traits for successful Pair Programming (PP). In PP two programmers are writing code side-by-side at the same workstation. Previous experience with PP is not required to take this survey.

The results of this survey will provide a basis for several experiments which will be conducted among Bowling Green State University Computer Science students during the 2005-2006 academic year. We appreciate the contribution that you make by taking this survey.

1. What is your primary job title?

- ☐ President/Vice President/Director
- ☐ Project Manager
- ☐ Software Developer
- ☐ Other

2. What is your gender?

- ☐ Male
- ☐ Female

3. What is your age?

- ☐ <30
- ☐ <40
- ☐ <50
- ☐ Greater than 50

4. How many years of experience do you have in the software development industry?

- ☐ >= 10 years
- ☐ >= 5 years
- ☐ >= 1 year
- ☐ Less than 1 year

5. How long have you been pair programming?

- ☐ >= 3 years
- ☐ >= 1 year
- ☐ >= 6 months
- ☐ none

6. What are the five most important traits that you would like your pair programming partner to have?

Personality Trait	Description
ACTIVE	Enthusiastic, dynamic, lively, agile, high energy, busy.
ADVENTUROUS	Try new things, take risks.
ATTENTIVE	Listen well, pay close attention to verbal and non-verbal messages.
CALM	Remain peaceful, composed, and serene under pressure.
COMPETITIVE	Strive to win, oppose a rival.
CREATIVE	Imaginative, inventive, original.
DECISIVE	Show firm determination in using judgment to make decisions.
DIPLOMATIC	Good communication skills, tactful, skillful at negotiating.
EMPATHIC	Compassionate, sensitive to or identify with the feeling of others, warm.
FLEXIBLE	Adaptable, versatile.
LOGICAL	Rational, analytical, reasonable.
OPEN-MINDED	Impartial, receptive to new ideas, free from prejudice or bigotry.
ORGANIZED	Structured, orderly, systematic.
OUTGOING	Sociable, friendly.
RESPONSIBLE	Dependable, trustworthy, reliable, loyal.

Please indicate the five personality traits that you think are most important for PP, ordered by the priority that they have for you. (1. most important, 2. second most important, and so on)

	1	2	3	4	5
Active	☐	☐	☐	☐	☐
Adventurous	☐	☐	☐	☐	☐
Attentive	☐	☐	☐	☐	☐
Calm	☐	☐	☐	☐	☐
Competitive	☐	☐	☐	☐	☐
Creative	☐	☐	☐	☐	☐
Decisive	☐	☐	☐	☐	☐
Diplomatic	☐	☐	☐	☐	☐
Empathic	☐	☐	☐	☐	☐
Flexible	☐	☐	☐	☐	☐
Logical	☐	☐	☐	☐	☐
Open-minded	☐	☐	☐	☐	☐
Organized	☐	☐	☐	☐	☐
Outgoing	☐	☐	☐	☐	☐
Responsible	☐	☐	☐	☐	☐

APPENDIX B. STUDENT PARTICIPANT CONSENT FORM

Research Participant Consent Form

You are invited to participate in a Master's Thesis study entitled "Personality Traits in Successful Pair Programming" conducted by Gulgunes Atli under supervision of Dr. Joseph Chao in the Computer Science department at Bowling Green State University.

The purpose of this study is to find out which personality traits are critical to successful pair programming.

This study will ask you to take one simple personality traits test. The test will take approximately fifteen minutes.

Information you provide will remain confidential and your identity will not be revealed. The result and information relating to the experiment may be published or used for educational purposes which may include lectures and journals. However, you will not be identified by name in any such publications or use.

This experiment does not involve any risk of harm to you other than the inconvenience of the time taken to participate which requires minimum amount of time.

Your participation is completely voluntary, and you are free to withdraw from the experiment at any time you choose, without any penalty.

Once the study is completed, you may request a summary of the results.

If you have any concerns about your treatment as a participant in this study, send an e-mail to Gulgunes Atli at gatli@bgsu.edu.

By completing and returning this form you are indicating your consent to participate in the "Personality Traits in Successful Pair Programming" study.

Participant's Signature: _____ Date: _____

If the participant is less than 18 years old;

Parent's Name: _____ Parent's Signature: _____

APPENDIX C. PAIR PROGRAMMING HANDOUT

Prior to the study the concept of pair programming was explained and handouts were provided to the participants.

Programming Assignments

For your assignments in this class, you are required to adopt a new strategy for software development: **pair programming**. This approach is very much at the forefront of software engineering practices today, and in many situations it has given impressive results. If producing better software, in half the time, while learning more, appeals to you, then you will enjoy pair programming!

Pair programming: what it is

As the name suggests, if you adopt this methodology, you will work with a partner. There is, however, much more to it than that:

- All of the work on the assignment will be done by both team members working together at a single PC at the same time. No work is done individually.
- At any given moment, one person is at the keyboard and is known as the *driver*. The other person is observing and is known as the *navigator*.
- The driver and navigator collaborate on all aspects of the software development: design, coding, debugging, etc.
- The driver does the actual typing of the code.
- The navigator is thinking about broader issues: How does the current code fit into the larger picture? How will we test the code to see if it works? What should we do next? Are we making any discoveries that require changes to code already written? The navigator also observes the code being written and checks it for defects, but please understand that the primary role of the navigator is not to say "Oh, I think you are missing a semicolon there." This role is called the navigator because the job is to steer the software development in the right direction.
- The driver and navigator are in constant communication: asking and answering questions of each other and debating the best way to solve the problem.
- Periodically, the driver and navigator change roles. This is a crucial part of the methodology, and if you choose to do pair programming, you are required to spend roughly equal amounts of time driving and navigating.

Pair programming: why it works

It is easy to see the benefits of the paired approach. As you have no doubt experienced, just "bouncing your idea off someone" is often very enlightening and very helpful. The act of explaining your solution helps crystallize it in your mind and helps you see any pitfalls. Add to that the fact that your partner will chime in with ideas of his or her own, and you can see that the power of two is much greater than the power of one.

Another great benefit often surfaces when you are working on some new aspect of programming. A common occurrence is simply getting stuck; that is, you arrive at some point where you don't know how to proceed or where your code is not working for reasons that are unclear. Sticking points like these can consume a few hours or even a few days.

Now imagine that you always have a partner to help you. In many cases you will find that where you are stuck, your partner will know the answer, and vice versa. Almost everyone who tries pair programming reports that there is far less "down time" due to programming difficulties.

Pair programming will also help you learn more. When your partner explains something to you, you learn something new. When you explain something to your partner, you learn it better. When you figure something out together, you both learn it, in less time than if you were working on your own. Mutual tutoring and problem solving is truly a win/win situation.

Pair programming: what it's not

Be sure you understand that pair programming does not mean dividing the assignment in half and each doing one part. That is one kind of software development team, but it is not what you will do in this class and it does not provide the benefits outlined above.

To reiterate: in pair programming, all the work is done with both team members at the same machine at the same time.

Pair programming: how to succeed

Compatibility: There are just a few of requirements for a successful pair programming project. In the first place, the partners must be compatible. Working with a friend is a really good idea; working with someone you don't like or normally disagree with is a bad one.

Schedule: Another key ingredient is compatible schedules. Since you have to work at the same time, you have to be available at the same time. When forming a team, be sure to discuss your schedules and be certain that you have adequate opportunities for working together.

Ability level: A final consideration has to do with ability levels. In any class, there is bound to be a range of programming abilities. Pair programming will work for you no matter where you are in that range, but it will work best if both partners are at about the same ability level.

Superstar: This last point deserves some thought. You might believe that the best thing to do is find a partner who is a superstar, and that if you do, then all your worries are over. Wrong! Two things tend to happen in those cases: one person does most of the work, and the other person fails the exams. You have to exercise some self discipline and realize that to learn what you are supposed to learn from the assignment, you need to put yourself in a situation where the work will be shared more or less equally.

How we will evaluate your work

The basic idea is simple: you submit one assignment for the team, and both partners receive the same grade.

There is one more aspect, though, that you need to be aware of. We may ask you to submit a brief "peer evaluation" or "pair report", on which you rate the performance of your team, your partner, and yourself.

This gives us confidence that the work was shared equally, and if it appears that it was not, we may ask you to come in and discuss the way your team worked. In rare cases, we might end up giving different grades to the two team members.

Pair programming: what to expect

Pair programming is currently the subject of much experimentation and evaluation. Based on results reported in other computer science departments, we think you will find that compared to solo programming, working in a pair means that:

- You will complete your assignments in roughly half the time. It is, of course, the same number of person-hours, but since you are working together, only half the calendar time is required.
- Your software will be cleaner, better organized, and have fewer bugs.
- You will learn more of the basic concepts that the assignment is designed to teach.
- Your exam performance will be about the same as if you had worked on your own.
- We think you will find pair programming to be a valuable programming and learning technique.

References

If you want to learn more about pair programming, a comprehensive discussion can be found in:

"Pair Programming Illuminated", by Laurie Williams and Robert Kessler, Addison Wesley, 2003.

The notion of pair programming was popularized as part of the Extreme Programming (XP) methodology. A good reference is:

"Extreme Programming Explained: Embrace Change", by Kent Beck, Addison Wesley, 2000.

How to piss off your pair

<http://c2.com/cgi/wiki?HowToPissOffYourPair>

APPENDIX D. PERSONALITY TRAITS TEST

The following is the personality traits test for determining the personality traits levels of the participants.

Personality Traits Test

1) Please enter your first and last name

2) In the following, there are phrases describing people's behaviors. Please use the rating scale below to describe how accurately each statement describes **you**. Describe yourself as you generally are now, not as you wish to be in the future. You will be assigned your pair programming partners based on your test result. Therefore, describe yourself as you honestly see yourself, in relation to other people you know of the same sex as you are, and roughly your same age. So that you can describe yourself in an honest manner, your responses will be kept in absolute confidence. Information you provide will remain confidential and your identity will not be revealed.

<i>Questions</i>	Very Inaccurate	Moderately Inaccurate	Neither Inaccurate nor Accurate	Moderately Accurate	Very Accurate
I carry the conversation to a higher level.	☐	☐	☐	☐	☐
I am open to change.	☐	☐	☐	☐	☐
I have difficulty imagining things.	☐	☐	☐	☐	☐
I rarely look for a deeper meaning in things.	☐	☐	☐	☐	☐
I am full of ideas.	☐	☐	☐	☐	☐
I am an original thinker.	☐	☐	☐	☐	☐
I don't pride myself on being original.	☐	☐	☐	☐	☐
I have no special urge to do something original.	☐	☐	☐	☐	☐
I believe in a logical answer for everything.	☐	☐	☐	☐	☐
I have no sympathy for criminals.	☐	☐	☐	☐	☐
I listen to my feelings when making important decisions.	☐	☐	☐	☐	☐
I believe that criminals should receive help rather than punishment.	☐	☐	☐	☐	☐
I take others' interests into account.	☐	☐	☐	☐	☐

I would never take things that aren't mine.	☐	☐	☐	☐	☐
I cheat to get ahead.	☐	☐	☐	☐	☐
I had a speeding ticket(s).	☐	☐	☐	☐	☐
When listening to a speaker I focus on his/her words, facial expressions, and body language.	☐	☐	☐	☐	☐
I pay attention to details.	☐	☐	☐	☐	☐
I daydream while listening to someone else's thoughts or ideas.	☐	☐	☐	☐	☐
If my phone rings while I'm having a serious discussion with someone, I answer it.	☐	☐	☐	☐	☐
I get excited by new ideas.	☐	☐	☐	☐	☐
I want to increase my knowledge.	☐	☐	☐	☐	☐
I don't think about different possibilities when making decisions.	☐	☐	☐	☐	☐
I prefer to stick with things that I know.	☐	☐	☐	☐	☐
I enjoy thinking about things.	☐	☐	☐	☐	☐
I love to think up new ways of doing things.	☐	☐	☐	☐	☐
I am not interested in theoretical discussions.	☐	☐	☐	☐	☐
I am not considered to have new and different ideas.	☐	☐	☐	☐	☐
I do things in a logical order.	☐	☐	☐	☐	☐
I listen to my brain rather than my heart.	☐	☐	☐	☐	☐
I sympathize with the homeless.	☐	☐	☐	☐	☐
I believe emotions give direction to life.	☐	☐	☐	☐	☐
I return extra change when a cashier makes a mistake.	☐	☐	☐	☐	☐
I appreciate people who wait on me.	☐	☐	☐	☐	☐
I try not to think about the needy.	☐	☐	☐	☐	☐
I pay my bills on time.	☐	☐	☐	☐	☐
I give my full attention if someone is talking to me.	☐	☐	☐	☐	☐
I can focus on my work and cannot be distracted easily.	☐	☐	☐	☐	☐
I fidget (play with hair, watch, pen, etc.) while listening to someone else's thoughts or ideas.	☐	☐	☐	☐	☐
I am uncomfortable if I'm not the one leading a conversation (i.e. choosing the topic, controlling the pace).	☐	☐	☐	☐	☐

APPENDIX E. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 1

Evaluation Sheet
Project #3 – Graphical Clock

Name(s) _____
Class time _____

Header comments complete (10)

- _____ (2) programmers' names
- _____ (2) class section, time, instructor's name
- _____ (2) program title
- _____ (2) description of the program
- _____ (2) description of the required input format

In-line Comments (5)

- _____ (3) comments on how the program calculates location of the clock hands
- _____ (2) comments on how the program writing the numbers inside the watch

Good programming Style (15)

- _____ (4) meaningful variable names
- _____ (3) white space usage
- _____ (4) indentation
- _____ (4) code is easy to read and understand

Usage of objects and member functions (15)

- _____ (3) Point class
- _____ (3) Line class
- _____ (3) Circle class
- _____ (3) Message class
- _____ (3) GraphicWindow

Interface design (10)

- _____ (2) user can easily understand what information should be entered and in what format
- _____ (2) hands in the watch are in different length and user can easily read the time
- _____ (2) formula to calculate the locations of numbers inside the watch
- _____ (2) adjustments of numbers to the watch
- _____ (2) clock design with minimum requirements

Correctness of the output (20)

- _____ (2) program calculates the end location of the hour hand correctly
- _____ (2) program calculates the end location of the minute hand correctly
- _____ (2) program calculates the end location of the second hand correctly
- _____ (3) program displays location of the hour hand correctly
- _____ (3) program displays location of the minute hand correctly
- _____ (3) program displays location of the second hand correctly
- _____ (5) program displays the numbers inside the watch

_____ **Total Points (75 points possible)**

APPENDIX F. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 2

Evaluation Sheet
Project #5 – Chutes and Ladders

Name(s) _____
Class time _____

Header comments complete (5)

- _____ (1) programmers' names
- _____ (1) class section, time, instructor's name
- _____ (1) program title
- _____ (2) description of the program

In-line and function comments complete (10)

- _____ (5) adequate in-line comments
- _____ (5) function documentation complete

Good programming Style (15)

- _____ (3) meaningful variable names
- _____ (2) white space usage
- _____ (2) indentation
- _____ (3) code is easy to read and understand
- _____ (5) there is not any global variables.
- _____ (5) Function main() is easy to read and describes the program clearly.

Function parameter and return value implemented correctly (25)

- _____ (5) RollDice function correct
- _____ (5) Move function correct
- _____ (5) CheckChutes function correct
- _____ (5) CheckLadders function correct
- _____ (5) CheckWin Function correct

PrintHeader, PrintOneMove (20)

- _____ (5) PrintHeader function used to print column headings
- _____ (5) PrintOneMove function used to print each move
- _____ (10) Data is correct and aligned neatly in columns

Correct output (20)

- _____ (5) Loop set up for each player to take a turn until one player wins
- _____ (5) Game stops when one player reaches or exceeds 81
- _____ (5) Name the game winner correctly
- _____ (5) Srand (time (NULL)) used only once correctly to seed the random number generator

_____ (100) Total Points

APPENDIX G. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 3

Evaluation Sheet
Assignment #3

Name(s) _____
Class time _____

Header comments complete (10)

_____ (5) programmers' names, class section, time, instructor's name, program title
_____ (5) description of the program

In-line Comments (10)

_____ (8) comments for all eight functions
_____ (2) other in-line comments

Good programming Style (10)

_____ (2) meaningful variable names
_____ (3) white space usage
_____ (3) indentation
_____ (2) code is easy to read and understand

Functions

Menu (10)

_____ (2) Displays a menu of the 7 choices
_____ (2) Displays them neatly according to the assignment sheet
_____ (2) Allows user to enter a selection
_____ (2) Function continues and not return until the user enters a valid selection
_____ (1) Quit menu item works properly
_____ (1) If the user enters invalid menu item, program gives a message to user

AddStudent(15)

_____ (2) Prompts the user to enter a name for a student
_____ (2) Calls FindStudent function
_____ (2) If the student is already on the list, prints a message stating the student is on the list and returns to the main function
_____ (3) If the student is not on the list, program prompts the user to enter the rest of the data for the new student.
_____ (3) Adds the student to the list
_____ (3) Prints a message saying the student was added then returns to the main function

SortStudents (10)

- _____ (2) Accepts list of students
 _____ (8) Sorts the list of students by name using bubble sort

ListStudents (10)

- _____ (4) Accepts list of students
 _____ (4) Prints the list of student's in a tabular form with column headers
 _____ (2) Data is neatly aligned

Classification (10)

- _____ (2) Accepts list of students
 _____ (3) Asks user to enter the classification as integer
 _____ (3) Prints out the students for that classification
 _____ (2) Data is neatly aligned

Deanslist (10)

- _____ (2) Accepts list of students
 _____ (8) Prints all of the students that have a GPA of 3.5 or better

Aboveaverage (10)

- _____ (2) Accepts the list of students
 _____ (3) Calculates the average GPA for the students on the list
 _____ (3) Prints the students that are above the average
 _____ (2) Data is neatly aligned

FindStudent (20)

- _____ (2) Accepts list of students
 _____ (2) Accepts student's name
 _____ (4) Searches the student on the list
 _____ (4) Returns the position of the student
 _____ (4) If the student is not on the list, returns -1
 _____ (4) Take cares of the situation when there is not any student on the list

_____ **Total (125 total)**

APPENDIX H. CODE QUALITY EVALUATION SHEET FOR EXPERIMENT 4

Evaluation Sheet
Assignment #5

Name(s) _____
Class time _____

Header comments complete (2)

- _____ (1) programmers' names, class section, time, instructor's name, program title
_____ (1) description of the program

In-line Comments (6)

- _____ (4) comments for all ten functions
_____ (2) other in-line comments

Good programming Style (5)

- _____ (1) meaningful variable names
_____ (1) white space usage
_____ (1) indentation
_____ (2) code is easy to read and understand

Menu (10)

- _____ (2) Displays a menu of the eight choices
_____ (2) Displays them neatly according to the assignment sheet
_____ (2) Allows user to enter a selection
_____ (2) Function continues and not return until the user enters a valid selection
_____ (1) Quit menu item works properly
_____ (1) If the user enters invalid menu item, program gives a message to user

AddStudent(15)

- _____ (2) Prompts the user to enter a name for a student
_____ (2) Calls FindStudent function
_____ (2) If the student is already on the list, prints a message stating the student is on the list and returns to the main function
_____ (3) If the student is not on the list, program prompts the user to enter the rest of the data for the new student.
_____ (3) Adds the student to the list
_____ (3) Prints a message saying the student was added then returns to the main function

Downsize (6)

- _____ (3) Accepts the list template
_____ (3) Removes every second student from the list

Dropstudent (10)

- _____ (2) Accepts the list template
- _____ (2) Asks for the name of a student
- _____ (3) Drops the student from the list
- _____ (3) Prints a message indicating the student was dropped, message is clear

SortStudents (10)

- _____ (2) Accepts list of students
- _____ (8) Sorts the list of students by name using bubble sort

ListStudents (6)

- _____ (2) Accepts list of students
- _____ (2) Prints the list of student's in a tabular form with column headers
- _____ (2) Data is neatly aligned

Classification (10)

- _____ (2) Accepts list of students
- _____ (3) Asks user to enter the classification as integer
- _____ (3) Prints out the students for that classification
- _____ (2) Data is neatly aligned

Deanslist (10)

- _____ (2) Accepts list of students
- _____ (8) Prints all of the students that have a GPA of 3.5 or better

Aboveaverage (10)

- _____ (2) Accepts the list of students
- _____ (3) Calculates the average GPA for the students on the list
- _____ (3) Prints the students that are above the average
- _____ (2) Data is neatly aligned

FindStudent (10)

- _____ (1) Accepts list of students
- _____ (1) Accepts student's name
- _____ (2) Searches the student on the list
- _____ (2) Returns the position of the student
- _____ (2) If the student is not on the list, returns -1
- _____ (2) Take cares of the situation when there is not any student on the list

_____ **Total (110 total)**

APPENDIX I. STATISTICAL ANALYSIS TABLES

Tables 1 thru 8 summarize the findings in ANOVA and unequal variance assumption tests. Numbers of pairs from each three groups, the average scores, and P-values under equal and unequal assumptions are presented. Tables 9 thru 16 summarize the findings in pair-wise Multiple Comparison Tests under equal and unequal variance assumptions. P-values under two assumptions are given for all three groups of pairs.

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	11	65.455	6.235	0.356	0.4550
	HL	10	68.100	3.542		
	HH	8	68.000	2.878		
Logical	LL	14	67.429	4.201	0.909	0.9241
	HL	8	66.500	6.188		
	HH	7	67.000	4.163		
Responsible	LL	9	64.444	5.434	0.123	0.1770
	HL	12	68.167	4.489		
	HH	8	68.375	2.875		
Attentive	LL	8	67.500	4.209	0.471	0.5074
	HL	10	68.200	3.994		
	HH	11	65.727	5.515		

Table 1: Results for Code Quality, Experiment 1

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	11	3.2273	0.5415	0.357	0.3499
	HL	10	2.9500	0.4048		
	HH	8	3.1563	0.3256		
Logical	LL	14	3.2143	0.4370	0.395	0.4622
	HL	8	3.0938	0.3995		
	HH	7	2.9286	0.5147		
Responsible	LL	9	3.0278	0.3411	0.803	0.7500
	HL	12	3.1458	0.5052		
	HH	8	3.1563	0.4989		
Attentive	LL	8	3.0625	0.5132	0.757	0.6992
	HL	10	3.2000	0.3291		
	HH	11	3.0682	0.5135		

Table 2: Results for Pair Compatibility, Experiment 1

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	6	91.667	3.141	0.758	0.5825
	HL	9	88.444	9.435		
	HH	5	88.000	13.304		
Logical	LL	10	88.00	9.67	0.424	0.2695
	HL	6	93.33	2.88		
	HH	4	86.50	12.77		
Responsible	LL	6	90.667	5.680	0.844	0.7994
	HL	7	87.714	9.793		
	HH	7	89.714	11.191		
Attentive	LL	4	91.50	5.20	0.808	0.7027
	HL	9	89.56	10.15		
	HH	7	87.71	9.79		

Table 3: Results for Code Quality, Experiment 2

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	6	6.083	1.758	0.085	0.0639
	HL	9	7.694	1.952		
	HH	5	8.550	1.304		
Logical	LL	10	7.875	1.987	0.582	0.5679
	HL	6	6.833	1.715		
	HH	4	7.188	2.304		
Responsible	LL	6	7.958	1.520	0.741	0.6666
	HL	7	7.179	1.852		
	HH	7	7.214	2.447		
Attentive	LL	4	6.250	1.837	0.123	0.1495
	HL	9	8.361	1.871		
	HH	7	6.893	1.682		

Table 4: Results for Pair Compatibility, Experiment 2

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	4	118.25	3.30	0.532	0.5878
	HL	5	119.40	5.68		
	HH	6	115.67	6.35		
Logical	LL	8	119.50	4.72	0.368	0.3639
	HL	3	115.33	4.04		
	HH	4	115.50	7.14		
Responsible	LL	4	119.75	3.30	0.583	0.5299
	HL	6	117.67	4.76		
	HH	5	115.80	7.46		
Attentive	LL	5	119.80	3.11	0.568	0.4374
	HL	4	116.50	5.74		
	HH	6	116.50	6.77		

Table 5: Results for Code Quality, Experiment 3

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	4	8.063	0.966	0.999	0.9975
	HL	5	8.100	0.518		
	HH	6	8.083	1.915		
Logical	LL	8	7.875	1.225	0.750	0.6511
	HL	3	8.083	2.005		
	HH	4	8.500	0.935		
Responsible	LL	4	7.750	1.708	0.828	0.8867
	HL	6	8.125	0.518		
	HH	5	8.300	1.690		
Attentive	LL	5	7.900	0.454	0.930	0.8839
	HL	4	8.125	1.548		
	HH	6	8.208	1.669		

Table 6: Results for Pair Compatibility, Experiment 3

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	4	96.25	3.59	0.176	0.0559
	HL	2	97.25	0.35		
	HH	5	99.60	1.52		
Logical	LL	5	97.80	3.11	0.820	0.7554
	HL	3	97.33	3.51		
	HH	3	98.83	1.89		
Responsible	LL	3	99.67	2.31	0.416	0.3019
	HL	4	96.75	1.89		
	HH	4	97.88	3.57		
Attentive	LL	3	99.67	2.31	0.491	0.4575
	HL	5	97.30	2.86		
	HH	3	97.33	3.06		

Table 7: Results for Code Quality, Experiment 4

Personality Traits	Pair Levels	# of Pairs	Avg. Score	Std. Dev.	P value (equal variance)	P value (unequal variance)
Open-minded	LL	4	7.875	1.714	0.433	0.4439
	HL	2	6.750	1.061		
	HH	5	5.950	2.521		
Logical	LL	5	5.350	1.973	0.098	0.1620
	HL	3	8.083	2.036		
	HH	3	7.917	0.520		
Responsible	LL	3	5.833	0.144	0.265	0.1218
	HL	4	8.188	1.675		
	HH	4	6.125	2.758		
Attentive	LL	3	7.00	2.165	0.532	0.6948
	HL	5	7.400	0.912		
	HH	3	5.583	3.503		

Table 8: Results for Pair Compatibility, Experiment 4

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.4051	0.4611
	LL-HH	0.4749	0.4704
	HL-HH	0.9989	0.9967
Logical	LL-HL	0.9011	0.9197
	LL-HH	0.9798	0.9706
	HL-HH	0.9780	0.9776
Responsible	LL-HL	0.1602	0.2369
	LL-HH	0.1842	0.1658
	HL-HH	0.9942	0.9883
Attentive	LL-HL	0.9470	0.9273
	LL-HH	0.6981	0.7104
	HL-HH	0.4599	0.4795

Table 9: Results for Code Quality, Experiment 1

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.3426	0.3932
	LL-HH	0.9372	0.9251
	HL-HH	0.5976	0.4677
Logical	LL-HL	0.8166	0.7915
	LL-HH	0.3649	0.4454
	HL-HH	0.7572	0.7735
Responsible	LL-HL	0.8303	0.7972
	LL-HH	0.8341	0.8119
	HL-HH	0.9986	0.9985
Attentive	LL-HL	0.8035	0.7851
	LL-HH	0.9996	0.9996
	HL-HH	0.7892	0.7599

Table 10: Results for Pair Compatibility, Experiment 1

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.7907	0.6140
	LL-HH	0.7942	0.8156
	HL-HH	0.9960	0.9976
Logical	LL-HL	0.4981	0.3073
	LL-HH	0.9572	0.9712
	HL-HH	0.4815	0.5645
Responsible	LL-HL	0.8391	0.7801
	LL-HH	0.9817	0.9773
	HL-HH	0.9160	0.9280
Attentive	LL-HL	0.9361	0.8873
	LL-HH	0.7964	0.6841
	HL-HH	0.9194	0.9221

Table 11: Results for Code Quality, Experiment 2

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.2211	0.2639
	LL-HH	0.0811	0.0518
	HL-HH	0.6653	0.6026
Logical	LL-HL	0.5732	0.5375
	LL-HH	0.8278	0.8572
	HL-HH	0.9584	0.9615
Responsible	LL-HL	0.7665	0.6865
	LL-HH	0.7846	0.7841
	HL-HH	0.9994	0.9992
Attentive	LL-HL	0.1551	0.1841
	LL-HH	0.8379	0.8301
	HL-HH	0.2653	0.2689

Table 12: Results for Pair Compatibility, Experiment 2

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.9481	0.9197
	LL-HH	0.7524	0.6858
	HL-HH	0.5203	0.5766
Logical	LL-HL	0.5015	0.3845
	LL-HH	0.4622	0.5935
	HL-HH	0.9991	0.9986
Responsible	LL-HL	0.8321	0.6969
	LL-HH	0.5542	0.5589
	HL-HH	0.8454	0.8760
Attentive	LL-HL	0.6570	0.5707
	LL-HH	0.5997	0.5525
	HL-HH	1.0000	1.0000

Table 13: Results for Code Quality, Experiment 3

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.9991	0.9970
	LL-HH	0.9997	0.9995
	HL-HH	0.9998	0.9996
Logical	LL-HL	0.9709	0.9821
	LL-HH	0.7286	0.6116
	HL-HH	0.9119	0.9383
Responsible	LL-HL	0.9023	0.9028
	LL-HH	0.8163	0.8772
	HL-HH	0.9747	0.9722
Attentive	LL-HL	0.9667	0.9548
	LL-HH	0.9253	0.8999
	HL-HH	0.9950	0.9969

Table 14: Results for Pair Compatibility, Experiment 3

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.8865	0.8393
	LL-HH	0.1654	0.2612
	HL-HH	0.5150	0.0478
Logical	LL-HL	0.9749	0.9779
	LL-HH	0.8842	0.8250
	HL-HH	0.8145	0.7894
Responsible	LL-HL	0.3860	0.2743
	LL-HH	0.6790	0.7114
	HL-HH	0.8329	0.8399
Attentive	LL-HL	0.5057	0.4716
	LL-HH	0.5827	0.5792
	HL-HH	0.9999	0.9997

Table 15: Results for Code Quality, Experiment 4

Personality Traits	Pairs	P-value (equal variance)	P-value (unequal variance)
Open-minded	LL-HL	0.8147	0.6123
	LL-HH	0.4021	0.4336
	HL-HH	0.8937	0.8281
Logical	LL-HL	0.1422	0.2956
	LL-HH	0.1713	0.1472
	HL-HH	0.9925	0.9894
Responsible	LL-HL	0.3164	0.1018
	LL-HH	0.9797	0.9741
	HL-HH	0.3516	0.4622
Attentive	LL-HL	0.9653	0.9487
	LL-HH	0.7111	0.8229
	HL-HH	0.5111	0.6755

Table 16: Results for Pair Compatibility, Experiment 4