A MULTIDIMENSIONAL APPROACH FOR ANALYZING VARIANTS OF CODE
WRITING QUESTIONS IN A CS1 COURSE

BY

LIIA BUTLER

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Tao Xie
Teaching Associate Professor Geoffrey Challen
ABSTRACT

To defend against collaborative cheating in code writing questions, instructors of CS1 courses with asynchronous exams can use the strategy of question variants, being manually written questions to be selected at random to assess the same learning goal. In order to create these variants, currently the instructors have to rely on intuition to accomplish the competing goals of ensuring variants are different enough to defend against collaborative cheating, and yet similar enough where students are assessed fairly. In this paper, we propose a multidimensional approach of analyzing these variants. We apply our approach on a dataset of 3 midterm exams from a large CS1 course. Our results show that (1) observable inequalities exist between variants and (2) these differences are not just limited to score. Our results also show that the information gathered from our analysis approach can be used to provide recommendations for improving design of future variants.
To my Grandpa Butler
ACKNOWLEDGMENTS

I would first like to thank Professor Tao Xie for his unwaivering support, guidance, and encouragement from the very beginning and whose invaluable advice and insights I have learned so much from.

I would also like to thank Teaching Associate Professor Geoff Challen, not only for his support and guidance in my research, but for also helping me become a better teacher during my teaching assistantships in CS 125 under his instruction.

I would like to thank all of the former/current members of the ASE group for all their help and support. I would also like to extend special thanks to Wing Lam and Angello Astorga for all the time they dedicated to mentoring me.

Lastly, I would like to thank my friends and family for always being there for me, even through the toughest of times. Particularly my parents, whose unconditional love and endless support made this all possible.
# TABLE OF CONTENTS

CHAPTER 1  INTRODUCTION ................................................. 1

CHAPTER 2  SUBMISSION STRUCTURE ...................................... 3
  2.1 Result Type ..................................................... 3

CHAPTER 3  DATA CONTEXT ................................................. 5
  3.1 Computerized Assessment Center .................................... 5
  3.2 Online Educational System ......................................... 5
  3.3 Exam Descriptions and Question Variants ........................... 6

CHAPTER 4  APPROACH ..................................................... 9
  4.1 End Outcome .................................................... 9
  4.2 Effort .......................................................... 9
  4.3 Struggle ....................................................... 10

CHAPTER 5  RESULTS ..................................................... 12
  5.1 End Outcome Results .............................................. 12
  5.2 Effort Results .................................................. 12
  5.3 Struggle Results ............................................... 13
  5.4 Overall Findings ............................................... 13

CHAPTER 6  DISCUSSION .................................................. 27

CHAPTER 7  RELATED WORK .............................................. 28
  7.1 Exam Content Analysis .......................................... 28
  7.2 Deriving Knowledge From Student Data ........................... 28
  7.3 Question Variants ............................................... 29

CHAPTER 8  CONCLUSION AND FUTURE WORK ............................ 30

REFERENCES .......................................................... 31
CHAPTER 1: INTRODUCTION

In introductory programming (CS1) courses, exams remain an integral part of assessing student knowledge. In recent years, the practice of paper-based exams has been disrupted by online assessments. While online assessments are not exclusive to CS1 courses, CS1 has seen particular benefit in offering online assessments. In paper-based exams, questions that require code writing to solve, in short as code writing questions, require tedious manual grading, which is prone to grading inconsistencies across student submissions. In contrast, in exams based on online assessments (in short as online exams), a code editor, compiler, and automated test cases not only take the grading burden off instructors but also provide consistent grading and feedback to students. Physical facilities (e.g., computerized assessment centers) designated for these online exams also lift the burden off instructors from the overhead of holding an exam during an instructional period while still having consistency in exam administration.

Asynchronous exams, where students are not taking a particular assessment at the same time, help address an assessment facility’s inability to support exams for a large class all at once. Asynchronous exams can be held over a given time period, and a student may choose a time slot during this period to take his/her exam. This form offers flexibility for a student to take an exam at a time most convenient for the student.

However, a major drawback of asynchronous exams is that they are more vulnerable to collaborative cheating. Collaborative cheating is when one student who has taken the exam supplies information about the exam to another student who has yet to take it. This information creates a clear advantage for the student going into the exam.

In order to defend against collaborative cheating, one exam design strategy is varying an asked question from student to student by means of question variants. Question variants are manually written questions that are put into a pool with other variants (typically two to four in total) and aim to assess the same learning goal. One variant is then randomly assigned to a student from the pool for the particular question. The student sees only that variant and is unaware of the existence of other variants. Figure 1.1 shows an example of the flow from question variants to the assigned question.

Although the idea of having multiple variants of a question is desirable, instructors who prepare question variants struggle to accomplish two competing goals: effectively defending against collaborative cheating (e.g., making the variants sufficiently different) and fairly assessing students’ same specific knowledge or skill (e.g., making the variants sufficiently similar). In particular, while creating sufficiently different variants for a question to realize
Figure 1.1: Linked List variant flow example

effective defense, the instructors need to ensure that the variants can maintain fairness of the assessment, regardless of the presence of collaborative cheating. For example, the variants should be sufficiently similar where each variant captures similar knowledge requirements needed to succeed at the question. To accomplish these two competing goals, the instructors currently have to rely on only gut feeling or intuition.

Although previous research [1] shows that providing question variants can help defend against collaborative cheating (i.e., the first goal), there exists no previous research on whether and how well question variants can accomplish fair assessment (i.e., the second goal). In practice, the instructors may check to see whether scores attained by students across question variants are roughly equal, but doing so does not offer sufficient insights on the extent of how question variants can impact student performance in CS1.

As the first attempt toward designing question variants to accomplish fair assessment in CS1 courses, in this paper, we propose a new multidimensional approach for analyzing question variants based on the set of submissions produced by all the students in a class for these question variants. Our approach first considers the overall distribution of the scores. Our approach then considers the distribution of time spent on a given variant. Finally, our approach breaks down the duration into the amount of time spent in a given student activity such as resolving compiler error and other actions in response to assessment feedback.

We apply our approach on the question variants from 3 midterm exams from the Fall 2018 semester of a large CS1 course. Our results show that (1) there are observable inequalities between variants, (2) variants whose scores are equal can still have significant differences in performance and effort required, indicating that relying on only scores may not be sufficient. Our results also show how information gathered from our analysis approach can be used to provide recommendations for improving design of future variants.
CHAPTER 2: SUBMISSION STRUCTURE

Each raw entry from the online educational system described in Section 3.2 is processed into a Submission Data Type representation (as shown in Figure 2.1). Each Submission has the question id, user id, the score received, the actual submission code, the timestamp of when it was submitted, and Result of submission (discussed in more detail in Section 2.1. We also store the feedback that the student receives (Checkstyle errors, compiler errors, stack traces from exceptions, failed assertions from test cases) for use in manual examination, but is not required as part of our analysis approach.

2.1 RESULT TYPE

Each Submission contains a Result categorizing the end result of the submission into one of five results: CheckstyleError, CompilerError, RuntimeException, FailedTestCase, and CorrectSolution. We provide a brief description of each result below:

- CheckstyleError - Submission contained code formatted not in compliance with a Checkstyle rule.
- CompilerError - Submission contained at least one compiler error.
- RuntimeException - An exception was thrown at runtime while tests were executing on the submission.
- FailedTestCase - Submission failed at least one test case.
- CorrectSolution - Submission was considered correct.

Exactly one Result is determined for each Submission by examining the score and parsing the grading results and feedback as stored in the raw entry.

```scala
data Submission = Submission {
    questionID :: String,
    userID :: String,
    score :: Double,
    code :: String,
    timestamp :: Integer,
    result :: Result
}
```

Figure 2.1: Submission Data Type
data Result = CheckstyleError | CompilerError
| RuntimeException | FailedTestCase
| CorrectSolution

Figure 2.2: Results Data Type
CHAPTER 3: DATA CONTEXT

We analyze the coding submissions from three 50-minute long midterm exams from the Fall 2018 semester of a large CS1 course taught in Java. The midterms are taken in a special computerized assessment center (Section 3.1) on an online homework and testing platform (Section 3.2). In addition to evaluating the correctness of code, the course also evaluates code quality by utilizing Checkstyle [2], a linting tool that statically checks code against a specified coding standard (e.g., limiting the number of characters on a single line of source code).

3.1 COMPUTERIZED ASSESSMENT CENTER

The computerized assessment center [3] allows students to asynchronously take computerized assessments in a controlled environment. Instructors of the course set a range of days when the assessment is open for students to take, typically 3-5 days. From that range, a student may then select any available time slot over the exam period to go to the facility and take the exam. Multiple courses can have overlapping exam periods and exams from different classes can occur in the facility at the same time.

The facility is a computer lab that supports over 80 students at a time. The facility takes measures to decrease cheating risks. Staff verify student identities, ensuring that correct students are at the correct time slot. Students must stow all belongings before heading to their randomly assigned seats. Staff then proctor once the examination period begins. In addition to the physical facility, the computers are also equipped with special control measures in attempt to prevent any unfair advantages including limited and controlled network and file system access, disabled access to removable storage (e.g., flash drives), and special screens on the monitors to prevent neighboring students from seeing what is on the screen.

3.2 ONLINE EDUCATIONAL SYSTEM

The course uses an online educational system [4] for both homework assignments and as the platform for the online assessments held in the computerized assessment facility. The system shows an overview of all questions in the assessment, and students can select which question to work on. The student can go to the next question from the previous question, or can go back to the overview and choose which question to work on next, so students do not have to go in the chronological order and can switch between questions freely. A student
can see the point value of the question and how many points the student has been awarded for the question at that time.

When a programming problem is presented, the interface displays the question, prompt written by the instructor, and a simple code editor with optional starter code. The student can save, or save and submit for grading. When the student submits for grading, the submission is sent to an external server for grading, compiled and run against a set of test cases, and the results are sent back; the process takes just several seconds. The test cases are instructor-defined for the question. These test cases consist of both a large number of randomly generated inputs and manually defined inputs for explicitly checking edge cases, depending on appropriateness for the specific question.

There is some available feedback returned to the student. The student can see the output from the compiler if there are a syntax error, stack traces from runtime errors, Checkstyle errors, and assertion errors for failed test cases run against the autograder (students only see the Assertion Exception stack trace with the expected value and the actual value in the test case, but they do not see the test input or code of the actual test case). The students are allowed unlimited attempts at programming problems.

3.3 EXAM DESCRIPTIONS AND QUESTION VARIANTS

In this section, we provide descriptions of the exams as well as a description of the questions and corresponding variants.

3.3.1 Exam 1

The first exam evaluated imperative programming skills. 827 students attempted the exam. The exam was 17 questions long, with 12 multiple choice, 1 free response, and 4 programming problems, presented in that order to students. 3 of the 4 programming problems were in variant format with 4 variants each and were the 14th, 15th, and 16th questions on the exam. Partial credit was not available on the programming questions. The question breakdown is as follows:

- E1Q1 Arrays - Ability to traverse an array
  - E1Q1V1 Even Array Sum - Sum even values in array
  - E1Q1V2 Odd Array Sum - Sum odd values in array
  - E1Q1V3 Even Index Array Sum - Sum values from even indices in array
E1Q1V4 Odd Index Array Sum - Sum values from odd indices in array

E1Q2 2D Arrays - Ability to traverse a 2D array
- E1Q2V1 Count Equal - Count all array values equal to reference value
- E1Q2V2 Count Not Equal - Count all array values not equal to reference value
- E1Q2V3 Count Greater Than - Count all array values greater than reference value
- E1Q2V4 Count Less Than - Count all array values less than reference value

E1Q3 String Parsing - Ability to parse a string
- E1Q3V1 Get Salary “:” - Get salary from record formatted “Name:Position:Salary”
- E1Q3V2 Get Salary “;” - Get salary from record formatted ”Name:Position:Salary”
- E1Q3V3 Get Yearly Salary “: ” - Get salary from record formatted “Name:Position:Weekly Salary” and multiply by a 50 week working year
- E1Q3V4 Get Yearly Salary “;” - Get salary from record formatted “Name:Position:Weekly Salary” and multiply by a 50 week working year

3.3.2 Exam 2
The second exam evaluated object-oriented programming skills. 735 students attempted the exam. The exam was 13 questions long with 10 multiple choice and 3 programming problems presented in that order to students. 2 of the 3 programming problems were written in variant format with each question having 4 variants each, and were the 11th and 12th questions on the exam. Partial credit was available on the programming questions. The question breakdown is as follows:

E2Q1 Class Design - Ability to create class, implement constructor, area method, and override equals method
- E2Q1V1 Equilateral Triangle - instance variable side length
- E2Q1V2 Rectangle - instance variables width and height
- E2Q1V3 Right Triangle - instance variable side
- E2Q1V4 Circle - instance variable radius
• E2Q2 Comparable Object - Ability to create class to extend Comparable, implement constructor, getter and setter methods, and compare method.
  – E2Q2V1 Dog - instance variable age
  – E2Q2V2 Cat - instance variable height
  – E2Q2V3 Turtle - instance variable speed
  – E2Q2V4 Ferret - instance variable length

3.3.3 Exam 3

The third exam evaluated knowledge over basic data structures and algorithms. 722 students attempted the exam. The exam was 16 questions long with 13 multiple choice questions and 3 programming questions. It had 2 variant-style programming questions with 4 variants each, and were the 14th and 15th questions on the exam. Partial credit was available on the programming questions. The question breakdown is as follows:

• E3Q1 Linked List - Ability to traverse a linked list
  – E3Q1V1 Count Even - Count number of items with even values
  – E3Q1V2 Count Odd - Count number of items with odd values
  – E3Q1V3 Count Positive - Count number of items with positive values
  – E3Q1V4 Count Negative - Count number of items with negative values

• E3Q2 Binary Tree - Ability to traverse a binary tree
  – E3Q2V1 Count Equal - Count number of nodes equal to reference value
  – E3Q2V2 Count Not Equal - Count number of nodes not equal to reference value
  – E3Q2V3 Count Greater Than - Count number of nodes greater than reference value
  – E3Q2V4 Count Less Than - Count number of nodes less than reference value
CHAPTER 4: APPROACH

In this section, we discuss how we use three different aspects of performance to form our approach of analyzing these variants. Since we compare variants in terms of performance in all points of the analysis, we do not include students who were assigned to a variant, but never attempted it. We provide the breakdown of overall students assigned and students who attempted, but we consider only the students who attempted the problem at least once during the analysis.

We define these dimensions based on the three qualities of a submission that offer distinct insight into the equivalence of these variants from a student performance perspective.

4.1 END OUTCOME

The first dimension we consider is the End Outcome. We consider this because this is what the student will be judged on in terms of final performance and is the most generally available information about a particular problem.

To put this aspect into concrete measures, we consider the overall distribution of the best scores of those who attempted a variant at least once. Since the best score that a student can receive on a question is discrete, we perform a chi-squared test with $\alpha$ value 0.05 and the null hypothesis that there is no difference between scores received from the variants.

4.2 EFFORT

The next dimension we consider is the total effort students put in. We consider this because we must consider variant equality from a problem-solving aspect. If there exist variants that are within the students’ overall skill set, but one variant is a more involved problem to solve, we should not consider these variants as equivalent.

For our evaluation, we will define effort as the total duration a user spent attempting to solve a problem.

We approximate duration in attempt to understand whether there are differences in performance among variants behind the final outcome. To approximate duration, we first order all submissions from a particular student for an exam. Then for a submission for question variant $q$ at index $i$, we name the timestamp for submission at $i-1$ the start time and the last submission for $q$ before a submission for a new appears the end time for that submission attempt series and calculate the difference. We repeat the process through the entire timeline.
and sum up the durations, should a student switch back and forth between problems. If a student attempts the variant before any other submissions, we use the timestamp of when the student began the exam as the start time. An example of this process is shown in Figure 4.1.

After we have durations for all variants of a question, we perform a one-way ANOVA to test the null hypothesis that the durations between these variants are equal. We again use $\alpha$ value 0.05.

4.3 STRUGGLE

The third dimension is the Struggle dimension. The Struggle dimension distinguishes itself from the Effort dimension in that we see where exactly the students put their efforts on a particular variant, and moreover potential sticking points present in a variant.

In addition to revealing insights of its own, this dimension can narrow the focus on why a variant performed the way it did in the End Outcome dimension and/or Effort dimension. This allows us to see past the students who might have been able to complete the problem in one try, and allow to see where other efforts were focused towards.

We define this aspect by breaking down the duration into a distribution of efforts during the time that a student spent on the question. We identify 5 types of student efforts during attempting a programming problem:

- First submission
- Resolving Checkstyle Error
- Resolving Compiler Error
- Resolving Runtime Exception
- Resolving Failed Test Case
We consider how long a student spent in each category and record the average percentages. This metric helps provide us with further insight on a particular sticking point on a particular variant compared to its alternatives, as well as providing quick insight on student behaviors toward a particular question as a whole. We also construct visual representations of students’ progression over time on a variant. For the visualization, we take the timeline of submissions for each student. We mark each submission with a dot corresponding to the result of that submission. The submission dot is followed by a line of the same color to show the time passed addressing the result. The white space preceding the first submission dot represents the amount of time until the first submission was submitted.
CHAPTER 5: RESULTS

We applied each of the dimensions in our approach to the data set described in Section 3. In this section, we describe the results and observations found from applying the approach.

5.1 END OUTCOME RESULTS

When we applied the End Outcome dimension of our approach onto the dataset, we found that one question, E3Q1, had a statistically significant difference in score among those who attempted the problem. We applied post-hoc Bonferroni tests to this question. We made comparisons between each of the 6 pairs of variants giving the Bonferroni-adjusted p-value of 0.0083 and found that there was statistically significant difference between variants 1 and 2, and between variants 2 and 3. We observe the common variant being variant 2, Count Odd. We will discuss the reasoning behind this difference in more detail in our case study of E3Q1 found in Section 5.4.1. Table 5.1 shows the full statistics and Tables 5.2 and 5.3 show all the calculated p-values.

5.2 EFFORT RESULTS

When we applied the Effort dimension of our approach onto the dataset, we found that four questions, E1Q1, E1Q3, E2Q1, and E3Q1, had a statistically significant difference in duration. We apply post-hoc Bonferroni tests to the questions. Since each question has 4 variants, we had 6 pairwise comparisons, so our Bonferroni-adjusted p-value for all question is 0.0083. For E1Q1, we found that there was a statistically significant difference in duration between V1 and V2, as well as V1 and V4. We see in Figure 5.1a one possible solution. However, in Figure 5.1b and Figure 5.1d, we see determining if a value is odd is involved in both. While more investigation is required, we can see a trend being suggested that checking for an odd value requires more problem-solving effort from the students’ perspectives.

For E1Q3, we found no statistically significant differences when comparing pairs of variants. While the additional step as seen in Figure 5.2c and Figure 5.2d could have added enough effort compared to Figure 5.2a and Figure 5.2b, there was not enough significance to determine an outlier.

For E2Q1, there was a statistically significant difference in duration between V2 and V4. We can see from Figure 5.3b that there is more involved with fulfilling the requirements with two member variables than one, which we hypothesize is the reason why we see it taking
occupying a longer duration. More investigation is required to concretely answer why V4 took on average the least amount of time.

Finally, for question E3Q1, we found there was a statistically significant difference in duration between V1 and V2, V2 and V3, and V2 and V4. We note that while all example solutions could potentially only have one to two differences, as seen in line 5 of each subfigure in Figure 5.4, the issue of determining if a value is odd does arise again and shows to be statistically significant.

Table 5.4 shows the average duration for all variants and Tables 5.5 and 5.6 show the p-values and pairwise p-values, respectively.

5.3 STRUGGLE RESULTS

We calculated the overall average percentage time spent on each of the 5 effort categories in a submission’s timeline. We see a fairly consistent distribution between the percentage time spent in a given effort between variants of a question throughout the questions.

However, we observe a notable discrepancy at the time spent on attempting to resolve failed test cases in E3Q1V2, Count Odd, versus all other variants. This result is particularly noticeable in Figure 5.5. We discuss the reasoning behind this in more detail in the case study in Section 5.4.1.

5.4 OVERALL FINDINGS

Overall, our approach was able to reveal that observable inequalities do exist among variants. We detected significant differences even at the first dimension in one question.

The second dimension revealed that, while having comparable scores, questions can still have variants that differ significantly in performance when duration is used as measurement. The same question with significant differences in score was observed to also contain significant differences in performance. We saw that variant 2, Count Odd, was statistically different than all three of the remaining variants. However, we see that the other even/odd question also had statistically significant differences in duration. In addition, we see that questions that contain variants with an extra component showed significant difference. However, only E2Q1 Class Design showed significant difference between a pair of variants, Rectangle and Circle.

We next present an in-depth case study of E3Q1 and show how we were able to produce recommendations on future variant designs from the information gathered in our analysis.
5.4.1 Case Study of E3Q1

Question E3Q1, Linked List, showed significant difference in both the first dimension and second dimension. We see that variant Count Odd is the variant that remains consistent in any pairwise significant differences. When we utilize the third dimension and examine the percentage time spent on effort and the actual graphs, we see that considerably more time and effort is spent resolving failed test cases. We suspect that this result is due to a misconception that exists with the “%” operator in Java. In Java, “%” is defined as the remainder operator, where the result takes the sign of the dividend, which is different than the mathematical modulus (i.e. remainder of Euclidean division), or other programming languages (e.g. Python), where the result takes the sign of the divisor. Students appeared to not internalize this distinction for Java. The difference between the two reveals itself when students have to check for odd values. Students who fall into the common pitfall of using “n % 2 == 1” to check for odd values fail any test cases where n is negative; therefore, the result of “n % 2” is -1. Consider the code snippets in Figure 5.6. In Figure 5.6a, we see the common pitfall of checking if the current item’s value modulo 2 is equal to 1.

Upon manual inspection of randomly selected submissions from this variant that resulted in a FailedTestCase result, we see that the pitfall is indeed present.

A student may be able to recover from this misconception. Consider a scenario where a student forgets Java’s implementation of taking the sign of the divisor, he or she may instead checking if the result not equal to 0, as seen in Figure 5.6b. Or a student may remember that he or she must check for both a positive and negative remainder and change their attempt to match the example in Figure 5.6c.

However, this misconception is not present in the other variants and is not the concept being evaluated in the question.

We provide our recommendation toward future questions that contain variants requiring use of the remainder operator but are not purposefully evaluating the ability to use the remainder operator. We suggest that these questions use the operator consistently across variants (e.g., testing for multiples of a given number).
1 public static int evenArraySum(int[] array) {
2     int sum = 0;
3     if (array == null) { return 0; }
4     for (int i = 0; i < array.length; i++) {
5         if (array[i] % 2 == 0) {
6             sum += array[i];
7         }
8     }
9     return sum;
10 }

(a) E1Q1V1 Example Solution

1 public static int oddArraySum(int[] array) {
2     int sum = 0;
3     if (array == null) { return 0; }
4     for (int i = 0; i < array.length; i++) {
5         if (array[i] % 2 != 0) {
6             sum += array[i];
7         }
8     }
9     return sum;
10 }

(b) E1Q1V2 Example Solution

1 public static int evenIndexArraySum(int[] array) {
2     int sum = 0;
3     if (array == null) { return 0; }
4     for (int i = 0; i < array.length; i++) {
5         if (i % 2 == 0) {
6             sum += array[i];
7         }
8     }
9     return sum;
10 }

(c) E1Q1V3 Example Solution

1 public static int oddIndexArraySum(int[] array) {
2     int sum = 0;
3     if (array == null) { return 0; }
4     for (int i = 0; i < array.length; i++) {
5         if (i % 2 != 0) {
6             sum += array[i];
7         }
8     }
9     return sum;
10 }

(d) E1Q1V4 Example Solution

Figure 5.1: E1Q1 Variant Example Solutions
```java
public int getSalary(String record) {
    if (record == null) { return -1; }
    String[] fields = record.split(" ");
    if (fields.length < 3) { return -1; }
    String salaryField = fields[2];
    int salary = Integer.parseInt(salaryField.trim());
    return salary;
}
```

(a) E1Q3V1 Example Solution

```java
public int getSalary(String record) {
    if (record == null) { return -1; }
    String[] fields = record.split(" ");
    if (fields.length < 3) { return -1; }
    String salaryField = fields[2];
    int salary = Integer.parseInt(salaryField.trim());
    return salary;
}
```

(b) E1Q3V2 Example Solution

```java
public int getSalary(String record) {
    if (record == null) { return -1; }
    String[] fields = record.split(" ");
    if (fields.length < 3) { return -1; }
    String salaryField = fields[2];
    int weeklySalary = Integer.parseInt(salaryField.trim());
    int yearlySalary = weeklySalary * 50;
    return yearlySalary;
}
```

(c) E1Q3V3 Example Solution

```java
public int getSalary(String record) {
    if (record == null) { return -1; }
    String[] fields = record.split(" ");
    if (fields.length < 3) { return -1; }
    String salaryField = fields[2];
    int weeklySalary = Integer.parseInt(salaryField.trim());
    int yearlySalary = weeklySalary * 50;
    return yearlySalary;
}
```

(d) E1Q3V4 Example Solution

Figure 5.2: E1Q3 Variant Example Solutions
public class EquilateralTriangle extends Shape {
    private int length;
    EquilateralTriangle(int l) {
        super("equilateraltriangle");
        length = l;
    }
    public double area() {
        return (Math.sqrt(3) / 4.0) * length * length;
    }
    public boolean equals(Object other) {
        if (other == null || !(other instanceof EquilateralTriangle)) {
            return false;
        }
        EquilateralTriangle otherTri = (EquilateralTriangle) other;
        return (length == otherTri.length);
    }
}

(a) E2Q1V1 Example Solution

public class Rectangle extends Shape {
    private int width;
    private int height;
    public Rectangle(int w, int h) {
        super("rectangle");
        width = w;
        height = h;
    }
    public double area() {
        return width * height;
    }
    public boolean equals(Object other) {
        if (other == null || !(other instanceof Rectangle)) {
            return false;
        }
        Rectangle otherRect = (Rectangle) other;
        return (width == otherRect.width && height == otherRect.height);
    }
}

(b) E2Q1V2 Example Solution

Figure 5.3: E2Q1 Variant Example Solutions
public class RightTriangle extends Shape {
    private double sideLength;
    RightTriangle(int sL) {
        super("righttriangle");
        this.sideLength = sL;
    }
    public double area() {
        return 0.5 * sideLength * sideLength;
    }
    public boolean equals(Object other) {
        if (other == null || !(other instanceof RightTriangle)) {
            return false;
        }
        RightTriangle otherRT = (RightTriangle) other;
        return (sideLength == otherRT.sideLength);
    }
}

(c) E2QIV3 Example Solution

public class Circle extends Shape {
    private int radius;
    Circle(int r) {
        super("circle");
        radius = r;
    }
    public double area() {
        return Math.PI * radius * radius;
    }
    public boolean equals(Object other) {
        if (other == null || !(other instanceof Circle)) {
            return false;
        }
        Circle otherCirc = (Circle) other;
        return (radius == otherCirc.radius);
    }
}

(d) E2QIV4 Example Solution

Figure 5.3 (cont.)

18
```java
protected int countEven() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value % 2 == 0) {
            count++;
        }
        current = current.next;
    }
    return count;
}
```

(a) E3Q1V1 Example Solution

```java
protected int countOdd() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value % 2 != 0) {
            count++;
        }
        current = current.next;
    }
    return count;
}
```

(b) E3Q1V2 Example Solution

```java
protected int countPositive() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value > 0) {
            count++;
        }
        current = current.next;
    }
    return count;
}
```

(c) E3Q1V3 Example Solution

```java
protected int countNegative() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value < 0) {
            count++;
        }
        current = current.next;
    }
    return count;
}
```

(d) E3Q1V4 Example Solution

Figure 5.4: E3Q1 Variant Example Solutions
<table>
<thead>
<tr>
<th>Question</th>
<th># Students Assigned</th>
<th># Students Attempted</th>
<th>% Attempted</th>
<th>Avg Score Attempted</th>
<th>Avg Score Attempted</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1Q1V1</td>
<td>182</td>
<td>182</td>
<td>100</td>
<td>93.4</td>
<td>93.4</td>
</tr>
<tr>
<td>E1Q1V2</td>
<td>225</td>
<td>223</td>
<td>99.1</td>
<td>88.4</td>
<td>89.2</td>
</tr>
<tr>
<td>E1Q1V3</td>
<td>197</td>
<td>195</td>
<td>99.0</td>
<td>92.9</td>
<td>93.3</td>
</tr>
<tr>
<td>E1Q1V4</td>
<td>224</td>
<td>219</td>
<td>97.7</td>
<td>87.4</td>
<td>89.0</td>
</tr>
<tr>
<td><strong>E1Q1 Total</strong></td>
<td><strong>827</strong></td>
<td><strong>819</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1Q2V1</td>
<td>194</td>
<td>187</td>
<td>96.4</td>
<td>84.5</td>
<td>87.7</td>
</tr>
<tr>
<td>E1Q2V2</td>
<td>210</td>
<td>203</td>
<td>96.7</td>
<td>86.7</td>
<td>89.7</td>
</tr>
<tr>
<td>E1Q2V3</td>
<td>218</td>
<td>217</td>
<td>99.5</td>
<td>84.9</td>
<td>85.3</td>
</tr>
<tr>
<td>E1Q2V4</td>
<td>205</td>
<td>201</td>
<td>98.0</td>
<td>88.3</td>
<td>90.0</td>
</tr>
<tr>
<td><strong>E1Q2 Total</strong></td>
<td><strong>827</strong></td>
<td><strong>808</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1Q3V1</td>
<td>203</td>
<td>154</td>
<td>75.9</td>
<td>43.5</td>
<td>56.5</td>
</tr>
<tr>
<td>E1Q3V2</td>
<td>210</td>
<td>182</td>
<td>86.7</td>
<td>51.9</td>
<td>59.9</td>
</tr>
<tr>
<td>E1Q3V3</td>
<td>203</td>
<td>155</td>
<td>76.4</td>
<td>37.4</td>
<td>49.0</td>
</tr>
<tr>
<td>E1Q3V4</td>
<td>211</td>
<td>163</td>
<td>77.3</td>
<td>38.9</td>
<td>50.3</td>
</tr>
<tr>
<td><strong>E1Q3 Total</strong></td>
<td><strong>827</strong></td>
<td><strong>654</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2Q1V1</td>
<td>189</td>
<td>187</td>
<td>98.9</td>
<td>91.1</td>
<td>91.9</td>
</tr>
<tr>
<td>E2Q1V2</td>
<td>178</td>
<td>176</td>
<td>98.9</td>
<td>90.8</td>
<td>91.8</td>
</tr>
<tr>
<td>E2Q1V3</td>
<td>188</td>
<td>186</td>
<td>98.9</td>
<td>93.5</td>
<td>94.5</td>
</tr>
<tr>
<td>E2Q1V4</td>
<td>180</td>
<td>178</td>
<td>98.9</td>
<td>93.8</td>
<td>94.8</td>
</tr>
<tr>
<td><strong>E2Q1 Total</strong></td>
<td><strong>735</strong></td>
<td><strong>727</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2Q2V1</td>
<td>181</td>
<td>179</td>
<td>98.9</td>
<td>81.8</td>
<td>82.7</td>
</tr>
<tr>
<td>E2Q2V2</td>
<td>177</td>
<td>176</td>
<td>99.4</td>
<td>81.5</td>
<td>81.9</td>
</tr>
<tr>
<td>E2Q2V3</td>
<td>187</td>
<td>185</td>
<td>98.9</td>
<td>80.4</td>
<td>81.3</td>
</tr>
<tr>
<td>E2Q2V4</td>
<td>190</td>
<td>188</td>
<td>98.9</td>
<td>81.6</td>
<td>82.4</td>
</tr>
<tr>
<td><strong>E2Q2 Total</strong></td>
<td><strong>735</strong></td>
<td><strong>728</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3Q1V1</td>
<td>177</td>
<td>173</td>
<td>97.7</td>
<td>88.1</td>
<td>90.2</td>
</tr>
<tr>
<td>E3Q1V2</td>
<td>192</td>
<td>187</td>
<td>97.4</td>
<td>82.7</td>
<td>84.9</td>
</tr>
<tr>
<td>E3Q1V3</td>
<td>181</td>
<td>177</td>
<td>97.8</td>
<td>89.6</td>
<td>91.6</td>
</tr>
<tr>
<td>E3Q1V4</td>
<td>172</td>
<td>170</td>
<td>98.8</td>
<td>89.9</td>
<td>90.1</td>
</tr>
<tr>
<td><strong>E3Q1 Total</strong></td>
<td><strong>722</strong></td>
<td><strong>707</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3Q2V1</td>
<td>178</td>
<td>175</td>
<td>98.3</td>
<td>82.8</td>
<td>84.2</td>
</tr>
<tr>
<td>E3Q2V2</td>
<td>163</td>
<td>159</td>
<td>97.5</td>
<td>80.6</td>
<td>82.6</td>
</tr>
<tr>
<td>E3Q2V3</td>
<td>194</td>
<td>191</td>
<td>98.5</td>
<td>84.9</td>
<td>86.2</td>
</tr>
<tr>
<td>E3Q2V4</td>
<td>187</td>
<td>181</td>
<td>96.8</td>
<td>81.2</td>
<td>83.9</td>
</tr>
<tr>
<td><strong>E3Q2 Total</strong></td>
<td><strong>722</strong></td>
<td><strong>706</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5.2: Question Score p-values for Attempted

<table>
<thead>
<tr>
<th>Question</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1Q1</td>
<td>0.213</td>
</tr>
<tr>
<td>E1Q2</td>
<td>0.406</td>
</tr>
<tr>
<td>E1Q3</td>
<td>0.197</td>
</tr>
<tr>
<td>E2Q1</td>
<td>0.682</td>
</tr>
<tr>
<td>E2Q2</td>
<td>0.743</td>
</tr>
<tr>
<td>E3Q1</td>
<td>0.013</td>
</tr>
<tr>
<td>E3Q2</td>
<td>0.161</td>
</tr>
</tbody>
</table>

Table 5.3: E3Q1 Score pairwise p-values for Attempted

<table>
<thead>
<tr>
<th>Variant pair</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1 vs. V2</td>
<td>0.0016</td>
</tr>
<tr>
<td>V1 vs. V3</td>
<td>0.2745</td>
</tr>
<tr>
<td>V1 vs. V4</td>
<td>0.5662</td>
</tr>
<tr>
<td>V2 vs. V3</td>
<td>0.0006</td>
</tr>
<tr>
<td>V2 vs. V4</td>
<td>0.0345</td>
</tr>
<tr>
<td>V3 vs. V4</td>
<td>0.1317</td>
</tr>
</tbody>
</table>
Table 5.4: Question Average Durations

<table>
<thead>
<tr>
<th>Question</th>
<th>Variant</th>
<th>Average Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1Q1</td>
<td>V1</td>
<td>04m 04s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>05m 54s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>04m 56s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>06m 02s</td>
</tr>
<tr>
<td>E1Q2</td>
<td>V1</td>
<td>05m 13s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>05m 20s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>05m 43s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>04m 56s</td>
</tr>
<tr>
<td>E1Q3</td>
<td>V1</td>
<td>11m 56s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>12m 07s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>13m 49s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>13m 32s</td>
</tr>
<tr>
<td>E2Q1</td>
<td>V1</td>
<td>08m 17s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>09m 01s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>08m 15s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>07m 33s</td>
</tr>
<tr>
<td>E2Q2</td>
<td>V1</td>
<td>09m 58s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>09m 57s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>10m 37s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>10m 34s</td>
</tr>
<tr>
<td>E3Q1</td>
<td>V1</td>
<td>05m 20s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>08m 05s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>05m 03s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>05m 06s</td>
</tr>
<tr>
<td>E3Q2</td>
<td>V1</td>
<td>09m 42s</td>
</tr>
<tr>
<td></td>
<td>V2</td>
<td>10m 51s</td>
</tr>
<tr>
<td></td>
<td>V3</td>
<td>09m 12s</td>
</tr>
<tr>
<td></td>
<td>V4</td>
<td>09m 32s</td>
</tr>
</tbody>
</table>

Table 5.5: Question Average Duration p-values

<table>
<thead>
<tr>
<th>Question</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1Q1</td>
<td>9.250E-05</td>
</tr>
<tr>
<td>E1Q2</td>
<td>0.272</td>
</tr>
<tr>
<td>E1Q3</td>
<td>0.019</td>
</tr>
<tr>
<td>E2Q1</td>
<td>0.049</td>
</tr>
<tr>
<td>E2Q2</td>
<td>0.747</td>
</tr>
<tr>
<td>E3Q1</td>
<td>6.213E-08</td>
</tr>
<tr>
<td>E3Q2</td>
<td>0.246</td>
</tr>
</tbody>
</table>
Table 5.6: Question Average Duration pairwise p-values

<table>
<thead>
<tr>
<th>Question</th>
<th>Variant pair</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1Q1</td>
<td>V1 vs. V2</td>
<td>0.0004</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V3</td>
<td>0.0529</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V4</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V3</td>
<td>0.0399</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V4</td>
<td>0.7840</td>
</tr>
<tr>
<td></td>
<td>V3 vs. V4</td>
<td>0.0144</td>
</tr>
<tr>
<td>E1Q3</td>
<td>V1 vs. V2</td>
<td>0.7974</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V3</td>
<td>0.0116</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V4</td>
<td>0.0399</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V3</td>
<td>0.0163</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V4</td>
<td>0.0539</td>
</tr>
<tr>
<td></td>
<td>V3 vs. V4</td>
<td>0.7176</td>
</tr>
<tr>
<td>E2Q1</td>
<td>V1 vs. V2</td>
<td>0.1584</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V3</td>
<td>0.9534</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V4</td>
<td>0.1471</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V3</td>
<td>0.1480</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V4</td>
<td>0.0077</td>
</tr>
<tr>
<td></td>
<td>V3 vs. V4</td>
<td>0.1687</td>
</tr>
<tr>
<td>E3Q1</td>
<td>V1 vs. V2</td>
<td>4.7794E-05</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V3</td>
<td>0.6189</td>
</tr>
<tr>
<td></td>
<td>V1 vs. V4</td>
<td>0.6828</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V3</td>
<td>2.8454E-06</td>
</tr>
<tr>
<td></td>
<td>V2 vs. V4</td>
<td>5.4979E-06</td>
</tr>
<tr>
<td></td>
<td>V3 vs. V4</td>
<td>0.9289</td>
</tr>
<tr>
<td>Question</td>
<td>% time on first submission</td>
<td>% time resolving Checkstyle Error</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>E1Q1V1</td>
<td>78.65%</td>
<td>03.35%</td>
</tr>
<tr>
<td>E1Q1V2</td>
<td>74.21%</td>
<td>03.18%</td>
</tr>
<tr>
<td>E1Q1V3</td>
<td>77.52%</td>
<td>02.81%</td>
</tr>
<tr>
<td>E1Q1V4</td>
<td>75.86%</td>
<td>03.49%</td>
</tr>
<tr>
<td>E1Q2V1</td>
<td>82.57%</td>
<td>01.91%</td>
</tr>
<tr>
<td>E1Q2V2</td>
<td>80.79%</td>
<td>02.67%</td>
</tr>
<tr>
<td>E1Q2V3</td>
<td>78.45%</td>
<td>02.83%</td>
</tr>
<tr>
<td>E1Q2V4</td>
<td>82.09%</td>
<td>01.22%</td>
</tr>
<tr>
<td>E1Q3V1</td>
<td>68.05%</td>
<td>01.68%</td>
</tr>
<tr>
<td>E1Q3V2</td>
<td>67.07%</td>
<td>00.92%</td>
</tr>
<tr>
<td>E1Q3V3</td>
<td>62.70%</td>
<td>01.31%</td>
</tr>
<tr>
<td>E1Q3V4</td>
<td>64.80%</td>
<td>01.13%</td>
</tr>
<tr>
<td>E2Q1V1</td>
<td>78.30%</td>
<td>01.27%</td>
</tr>
<tr>
<td>E2Q1V2</td>
<td>73.19%</td>
<td>02.67%</td>
</tr>
<tr>
<td>E2Q1V3</td>
<td>76.52%</td>
<td>01.53%</td>
</tr>
<tr>
<td>E2Q1V4</td>
<td>76.66%</td>
<td>01.07%</td>
</tr>
<tr>
<td>E2Q2V1</td>
<td>65.73%</td>
<td>02.55%</td>
</tr>
<tr>
<td>E2Q2V2</td>
<td>65.48%</td>
<td>02.51%</td>
</tr>
<tr>
<td>E2Q2V3</td>
<td>63.45%</td>
<td>02.46%</td>
</tr>
<tr>
<td>E2Q2V4</td>
<td>63.32%</td>
<td>02.79%</td>
</tr>
<tr>
<td>E3Q1V1</td>
<td>78.14%</td>
<td>01.00%</td>
</tr>
<tr>
<td>E3Q1V2</td>
<td>60.83%</td>
<td>00.73%</td>
</tr>
<tr>
<td>E3Q1V3</td>
<td>78.49%</td>
<td>01.23%</td>
</tr>
<tr>
<td>E3Q1V4</td>
<td>81.11%</td>
<td>00.83%</td>
</tr>
<tr>
<td>E3Q2V1</td>
<td>66.59%</td>
<td>01.39%</td>
</tr>
<tr>
<td>E3Q2V2</td>
<td>61.04%</td>
<td>01.28%</td>
</tr>
<tr>
<td>E3Q2V3</td>
<td>66.71%</td>
<td>03.93%</td>
</tr>
<tr>
<td>E3Q2V4</td>
<td>65.12%</td>
<td>01.10%</td>
</tr>
</tbody>
</table>
Figure 5.5: Timelines for question E3Q1
protected int countOdd() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value % 2 == 1) {
            count++;
        }
        current = current.next;
    }
    return count;
}

(a) Incorrect example with pitfall

protected int countOdd() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value % 2 != 0) {
            count++;
        }
        current = current.next;
    }
    return count;
}

(b) Example Correction 1 with one comparison

protected int countOdd() {
    int count = 0;
    Item current = start;
    while (current != null) {
        if (current.value % 2 == 1 || current.value % 2 == -1) {
            count++;
        }
        current = current.next;
    }
    return count;
}

(c) Example Correction 2 with two comparisons

Figure 5.6: E3Q1V2 Pitfall and Correction Code
CHAPTER 6: DISCUSSION

While only requiring information extractable from submission records does allow our approach to be more generally applicable, external and human factors are not captured in this analysis, with two major limitations.

First, since our analysis requires submissions from a student to analyze, and thus we exclude those variants that did not have any submission, we do not know whether an excluded variant had influence on why a student did not submit. For example, a student could have looked at the prompt and deliberately skipped it, ran out of time, and did not even see the question, or even attempted to solve the question and simply forgot to submit.

Second, our duration approximation is limited. We made the decision that the start time begins at the end of the previous question submission or when the student begins the exam. This decision naturally makes the assumption that students immediately begin focusing on the next question, while during the exam, there may have been other events that happen before the student begins the question, such as deciding which question to attempt next or first. Likewise, duration between attempts for the same question considers all time that was spent on the next attempt, while there may have been other occurrences unaccounted for, such as a student’s thinking about a previous question while still working on a current question. This decision would lead our analysis to overestimate duration, particularly for when students first begin a question, as more time is likely to pass during this period that we assume the student is working on the attempt. Likewise, students may be addressing issues other than/in addition to the issue as indicated in the submission state that we use in the Struggle dimension.
CHAPTER 7: RELATED WORK

To the best of our knowledge, our work is the first to study the problem of fair assessment with question variants in the CS1 course setting. Our work builds on a number of subareas regarding student assessment. We address the subareas most relevant to our work and discuss the related work in the following subsections.

7.1 EXAM CONTENT ANALYSIS

A number of works have taken a more critical look into instructor-created assessments and the validity of the assumptions instructors make when creating these exams in both inside the CS domain [5, 6, 7] and expanding to other subjects [8]. However, these works look at exams more holistically and do not consider the finer grain of variants of a particular question.

7.2 DERIVING KNOWLEDGE FROM STUDENT DATA

There exists related literature that also attempts to understand assessment components using data generated, such as using student data to gain better insight about the assessment process in CS settings [9].

There is work on using data generated from educational systems to measure similarity between educational items [10, 11]. While these works consider correctness and incorrectness, as well as response time, item similarity for adaptive educational systems does not have the same application for detecting open-ended programming problem difference. Educational system items for a particular knowledge component are intended to be similar and fairness is not a factor, so there is not the motivation of the question writer to find a balance of similar and different as for an exam setting. In addition, the items evaluated were from simple questions with one attempt from domains such as basic arithmetic and grammar. This is vastly different than open-ended programming questions with unlimited answers that can exhibit correct behavior and unlimited attempts, where more context about the submission’s evolution is necessary.
7.3 QUESTION VARIANTS

Analysis has been done regarding open-ended question variants has been done for English essays in high stakes exams [12]. The premise is similar to coding question variants, where several variants are derived from a base question. However, the differences between the exams and subjects are far too great for the analysis to be appropriate to CS1 settings. In addition, the essay questions are in a controlled timed setting, separated from the rest of the exam, while the time allocated in the CS1 assessments is for all parts of the exam.
CHAPTER 8: CONCLUSION AND FUTURE WORK

We have presented an approach for analyzing variants of code writing questions in a CS1 course setting. We use three dimensions to analyze these variants: End Outcome, Effort, and Struggle. We found that there exist observable differences among the variants. Moreover, we showed that questions that may be considered equivalent in terms of score could still contain differences in terms of student effort. Finally we used a case study from applying our approach onto a dataset of CS1 exam question variants to show that results from our analysis approach can be used to provide recommendations for improving future variants.

While our work offers the first step, there is still much more to be done in future work to understand variants of code writing questions. Certain questions can be considered equivalent, but we do not know how effective they are at defending against collaborative cheating. The natural next step would be to attempt to understand how effective these variants were in defending collaborative cheating. When we sufficiently understand defense effectiveness and fair assessment, respectively, and the relationship between them, we can begin to construct best practices for designing these variants.
REFERENCES


