Work In Progress - Programming Assignment Summary for Analysis, Qualitative Assessment and Continuous Improvement in CS1 – CS3

Karina Assiter
Wentworth Institute of Technology, assiterk@wit.edu

Abstract - **Homework assignments are essential for skills development in a first or second year programming course (CS1 – CS3), and instructors who assign homework in CS1 – CS3 would like submitted solutions to accurately reflect student performance. Unfortunately, this is hard to gage when students increasingly have access to solutions on the internet and are frequently encouraged to collaborate. To address these concerns, we incorporated a *summary* into the list of programming assignment requirements. Though it serves many roles, including as a vehicle for analysis of course topics, it's most important role is as qualitative measure of an individual students' performance *(which indirectly informs their quantitative assessment).* This paper describes this *assignment summary*: including its original purpose, example analysis and summary questions, textual analysis of student responses, and preliminary correlations between responses and student assignment scores.**

Index Terms - Assessment, Computer Science, CS1 – CS3, Programming.

INTRODUCTION

In the CS1 – CS3 foundation courses, students develop basic software development skills through programming based homework. As instructors, we would like submitted solutions to accurately reflect student performance, and to indicate the level of student understanding of course topics. This is difficult to gage when students are able to access software solutions from internet repositories, and are frequently encouraged (for sound pedagogical reasons) to work in groups or pairs. Thus, graders who assess submitted work have to evaluate: how much of it is original, and how much did an individual student (in a group or pair) contribute to the final solution.

I. Plagiarism

Though most students want to solve their own programming problems, for every assignment there are likely to be student (sub) solutions acquired from an electronic source, or from another person; as the author states in [1] "the store of solution code out on the internet keeps getting better. The search engines are getting better at finding it". Numerous authors highlight both the methods in which students cheat, as well electronic and traditional methods of detecting plagiarism [2]-[4].

II. Fuzzy Collaboration

A more complex problem is assessing individual performance when students collaborate on assignments [5]. There is increasing evidence that students benefit from peer-based collaboration [6] - [7]. Unfortunately, there is a fine line between *collaboration*, and *cheating*, where either a) one (or more) student(s) don't contribute their share, or where, b) students alternate assignments. This type of cheating, which we term "fuzzy collaboration", is hard to detect; we were only able to discover it in cases where a student admitted to it verbally, or when one of the students (in a team or pair) had little or no understanding of the problem concepts.

ASSIGNMENT SUMMARY

To address our assessment concerns, we incorporated a *summary* into the list of programming assignment requirements. Though it serves many roles, including as a vehicle for analysis of course topics, it's most important role is as qualitative measure of an individual students' performance, which can then indirectly inform their final quantitative assessment.

Though we were initially concerned with evaluating student performance, we discovered that the summaries served many purposes:

- Requiring Analysis of the assignment topic
- Confirming understanding of assignment concepts.
- Verifying that partners each contributed
- Evaluating the complexity of an assignment
- Evaluating work for plagiarism
- Gathering feedback about an assignment for the future.
- Strengthening relationship between instructor and student

978-1-4244-6262-9/10/$26.00 ©2010 IEEE

Summary questions vary depending on the course and the assignment topic; examples are shown in Table I.

TABLE I
SUMMARY QUESTIONS

CS2 A3 – Social Networking
1. Describe how you approached and solved the problem. If you worked in pairs, how did you make sure that each student contributed? **2.** Where did you have trouble? How did you move forward? What topics still confuse you? **3.** What did you learn from this assignment? (be specific)
CS3 A1 - Review of Java & OOP
1. Why couldn't you add a main method to the Region class to test it before proceeding? **2.** Could we have added Generics to this assignment? If so, how? If not, what would have been a better example for generics? **3.** Where did you have trouble? How did you move forward? What topics still confuse you? **4.** What did you learn from this assignment? *(Please be specific)* 5. Any suggestions for improving this assignment in the future?

STUDENT RESPONSES

In this section we divide student responses into subsections based on summary question. Due to space constraints, only a small set of illustrative examples are included from our complete set of student responses.

I. Describe how you approached and solved the problem

Well ***I guess*** I took it step by step of the criteria, making sure each piece would work. Read through the book and example source codes. **(58)**
I simply started writing out the classes in the order presented in the assignment and continued working ***until I got sick of doing so*** **(63)**
To begin the project, Frank made a skeleton of the classes and all of the objects inside of them. We did this to be even simpler than a UML Class Diagram. This skeleton had just the names of the classes and their objects (specifying if it was an array or not), but neither their types nor the accesors or mutators. Next, I went into the main class (Friendbook) and plotted out the basic methods that would be needed to complete our project. Inside these, I placed no actual code other than printing messages such as "Please input username". This mapped out that code we would need to do later, without actually requiring "real" code. Next, Frank made the other classes and the methods (constructors, mutators, and accesors) that were needed for them to function. After this was done, I went in and ***wrote the code that was needed for the main class to perform its functions properly.(100)***

We've observed that student scores increase with response *length*. In addition, the response *content* provides an indication of the students' perception of their performance. For example, "I guess" suggests a lack of problem/solution understanding, while "wrote the code ... (sic) for the main class to perform its functions properly" suggests that the student submitted a fully functioning assignment.

II. Where did you have trouble? How did you move forward? What topics still confuse you?

I had a lot of trouble implementing the different methods. Comment and Wall were particularly difficult. **Much of the syntax still confuses me.** I looked for help from upperclassmen and got many questions answered. **(58)**
At first I had a flawed vision of how object oriented programming actually worked. I was unaware of how to properly integrate different classes together. After this assignment (and the research that entailed trying to solve it) ***I now have a much better understanding of how it all works.*** In the future I'll have no trouble with this. **(100)**

In general, when students had high scores their responses emphasized concepts learned; otherwise, they tended to focus on their problems.

III. What did you learn from this assignment?

I learned a lot about fixing errors. I also learned how to make the array for the Wall. I'm ***learning*** how to make coding ***simpler*** and ***not as long and unnecessary***. **(58)**
In my research for this project, one of the things I learned about was the idea of exceptions. Specifically, the exception of when an array goes out of its set parameters. I had very little idea of them before the assignment. As I said before, ***I now know how to integrate classes together. I feel as if this is the assignment that has helped me understand the subject material the most.*** **(100)**

Surprisingly, students always indicate that they've learned something even when they receive a low score.

CONCLUSION

It would be a challenge to make verifiable conclusions based on all questions from every class. However, we have found patterns in responses that could be used as a guide for assessing student work, including:

- Scores not always indication of concepts learned; even when they can't get their programs functioning, students frequently indicate that they understand components of assignment concepts.
- Response length varies in expected way with question type and with prior experience. For example, more experience equals shorter response to: "what did you learn?"
- Response content provides indication of students' perception of their performance.
- Where students *received high scores*, they emphasized concepts learned; otherwise, they focused on their problems, and their own weaknesses.

FUTURE WORK

As this is a work-in-progress, the primary question still to be answered is whether or not we could detect plagiarism with assignment summaries. As this set of summaries did not reveal a plagiarized solution, we couldn't answer this question definitively. In addition to evaluating the summary as a first-step in detecting plagiarism, we intend to: evaluate question variations for the amount of detail they elicit from student responses, and request student feedback on their perceptions of the assignment summary.

978-1-4244-6262-9/10/$26.00 ©2010 IEEE

REFERENCES

[1] Parlante, N., "Cheating and the Internet". *inroads - SIGCSE Bulletin,* Vol. No 39, December 2007, 29 - 30.

[2] Daly, C. and Horgan, J., Patterns of Plagiarism*, SIGCSE'05*, 2005, 383 - 387.

[3] Harris, J.K., "Plaigiarism in computer science courses"*, Proceedings of the Conference on Ethics in the computer age*, 1994, 133 - 135.

[4] Mann, S. and Frew, Z., "Similarity and originality in code: plagiarism and normal variation in student assignments"*, Proceedings of the 8th Australian conference on computing education,* 2006, 143 - 150.

[5] Carter, J., "Collaboration or Plagiarism : What happens when students work together"*, iTiCSE '99*, 52 - 55.

[6] Baer, J., "Grouping and Achievement in Cooperative Learning", *College Teaching*, 2003, Vol. No 51, 169 - 174.

[7] Werner, L., Hanks, B. and McDowell, C. , "Pair programming helps female computer science students"*, Journal on Educational Resources in Computing*, Vol. No 4, 2004.

[8] Joyce, D., "Raising awareness about academic integrity", *SIGCSE Bulletin,* Vol. No 38, September 2006.

[9] Dick, M, et al. "Addressing Student Cheating: Definitions and Solutions", *SIGCSE Bulletin,* Vol. No 35, 2003, 172 - 184.