# Second Chances: Does Allowing Homework Resubmissions Affect Student Learning?

R. Pito Salas
Brandeis University, Waltham, Massachusetts, USA
pitosalas@brandeis.edu

***Abstract*—In our quest to create conditions for greater and more powerful learning for our students, we have experimented with giving students a chance to resubmit programming assignments. The resubmission option would come after they receive a grade and feedback on their original submission. This paper describes a quantitative study of an introductory Java programming course at Brandeis University where we offered the resubmission option. We show a particular approach to collecting and analyzing the data and reach preliminary conclusions about the impact of resubmissions on learning outcomes.**

***Index Terms*—assessment, resubmission, assignments, computer science education**

## I. Introduction

In our quest to create conditions for greater and more powerful learning for our students, we have experimented with giving students a chance to resubmit programming assignments. The resubmission option would come after they receive a grade and feedback on their original submission.

Our intuition is that by giving students a reason to actually apply what they learn from the initial grade, they will deepen their understanding of what we are trying to get across, and thereby create a more effective teaching experience. This study is our attempt to measure and demonstrate the effect of this policy.

In the Computer Science department at Brandeis University, all Computer Science majors need to take a 2-semester sequence of introductory programming courses, known as Cosi11a and Cosi12b. Like many, Brandeis teaches the Java programming language in the introductory courses.

Naturally there is high demand and so the teaching load is carried by a rotating cast of five professors. These courses are specifically focused on teaching students to program, in our case in Java. In other words, the primary focus is teaching the practice and skill of programming, much less so on theoretical aspects of Computer Science.

Students are assigned a series of programming assignments, coordinated with what is being taught that week. The students consider these assignments to be the most challenging aspect of the course. They are also the most important learning experiences for the students and carry the greatest grading weight.

When considering the notion of resubmission of assignments, one needs to consider several dimensions in the policy space. In general policies like these have obvious or sometimes subtle impact on student motivation and behavior, which will be considered later.

- **Count:** How many times can the student resubmit. Not at all, once, twice, or unlimited?
- **Optionality:** Are students permitted, encouraged, or required to resubmit?
- **Grading:** How does the grading of the resubmission affect the overall grade? Does it replace it, or do we take an average, or something else? Can the final grade ever be reduced by resubmitting?
- **Time frame:** How much time is there for resubmission? Does it need to occur within a certain number of days of the original due date or is it open ended?

## II. Literature Review

This is not a new idea. We encounter it in connection with Automatic Assessment which permit resubmissions. Karavirta et al [2] are interested in understanding the clusters of student behaviors and decisions when it comes choosing to resubmit. By using data generated by the Automatic Assessment tool they are able to derive fine grained quantitative insights, which lead to a set of clusters with catchy names reflecting the authors guess of the students learning style, motivation and skills.

Holland-Minkley and Lombardi [1] are able to compare outcomes between two semesters of the same course pre- and post- resubmission policy. They make the interesting observation that the practice of allowing resubmissions in a programming course is in some ways analogous to the practice integration of code reviews into courses. Holland-Minkley et al use an interesting decision tree analysis to find patterns of behavior.

The wide variation in times students reported to complete each PA, while surprising, is consistent with the the wide range of programmer productivity described anecdotally as "10x Programmers", and in the literature, for example in Parnin and Peitek [3], who says: "But even after extensive experience in programming, there are still tremendous differences in programmer expertise: So called **10xers** can be 10 or more times as productive as other programmers who have spent an equal amount of time with programming Oram and Wilson (cite10x and Sackman and Grant [4])."

## III. Background

### A. Programming Assignments

This study is focused on "Programming Assignments," which we will refer to as **PA**s throughout the rest of this paper.

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

There were 11 **PAs** during the semester of the study referred to as PA1 though PA11. Some but not all of the textbfPAs contain starter code. Each PA package is in the form of an archive containing:

- A multi-page document, which contains a detailed description of the problem, with relevant Computer Science concepts, a step-by-step set of instructions, and the required deliverables.
- Sometimes there will be starter Java source code, so the student is steered in the right direction. Other times there is no starter code, which challenges the student to structure the solution from scratch.
- Sometimes there are a series of approximately 5 unit tests. Students are reminded that the set provided are not comprehensive and that successful students know to write additional tests.

Here is a more detailed enumeration of the PAs:

| # | Name | Comments | Hours |
|---|---|---|---|
| 1 | Warmup Problems | Basic algorithms requiring looping, no classes | 9.2 (6.1) |
| 2 | Game of 15 | Working with arrays | 5.5 (3.4) |
| 3 | Caesar Cipher | Encrypt and Decrypt with a common algorithm | 7.9 (4.2) |
| 4 | Snow Day | Fairly complicated simulation of snow trucks on roads | 13.0 (6.6) |
| 5 | Stats | Compute statistics over various kinds of collections | 7.4 (3.9) |
| 6 | Timing Attack | Use of external API and timing of code | 7.6 (4.0) |
| 7 | Knowledeg Base | Complicated inheritance scenario | 8.1 (5.4) |
| 8 | Graphs | Representing graphs and graph algorithms | 10.9 (5.6) |
| 9 | Twitter Cloud | External APIs to Twitter and other services | 8.4 (4.7) |
| 10 | Pongout | Simple Pong-like game using Processing | 8.1 (5.4) |
| 11 | HootScript | Tokenizing, parsing and executing | n/a |

Fig. 1: Programming Assignments

**Important:** The final column is from students' self-reported hours worked to complete the assignment. It is apparent that the degree of difficulty of the assignments varied fairly widely and that we were not exactly successful in having a nice graduated series from relatively easier to relatively harder. The high standard deviations could be seen as consistent with the so-called **10x Programmer** effect mentioned in [3]

### *B. Submission and grading Procedures*

The protocol for processing and grading each of the PAs is as follows:

1) Students receive the assignment consisting of an archive made available via a download from our course management system. See the section above for more details on what is in the assignment package.
2) Students work on the assignment for up to a week, writing the programs and running the unit tests. They are reminded that the tests provided are not comprehensive and that they should write additional unit tests.
3) The student submissions are subjected to an automated test suite (which is far more comprehensive than the one they received). The percentage of failing tests will form 0.7 of the grade for the assignment.
4) Students meet one-on-one with Teaching Assistants (TAs) for an in person code review. The students receive the feedback while the TA arrives at a qualitative score which will count as the other 0.3 of the grade for the assignment.
5) The students next have the option during the following week, to submit the same assignment again, at which point we go through the same process to produce a resubmission score. The final score for the PA is simply the average of the initial and resubmission score. The PA then receives a final-score, the average of the initial and resubmission score.

Our policies were that students could (but didn't have to) resubmit, at most once, within a week of the initial deadline. The grade was computed as the average of the initial and resubmission score, if any.

### *C. Students' subjective feelings*

We were also interested in how the students felt about the resubmission policy. After all, it had its pros and cons.

On the one hand, it affords the student a chance to apply what they have learned from both the automated tests as well as the individual code review and feedback. Idealistically, they might have been doing this to learn to be better programmers. But also, they saw this as simply a chance to improve their grade.

On the other hand, it added work to their already full plate. In effect during any given week they were both responsible for this week's assignment, and last week's assignment's resubmission. And with more work came more stress and pressure.

Yet when queried as a group, with a multiple-choice question administered in class, 99% of the students recommended not changing the policy.

They overwhelmingly appreciated the chance to resubmit.

## IV. Questions

At the heart of the research is ascertaining whether and to what extent a resubmission protocol can improve learning outcomes. This immediately begs the question, what is a learning outcome and how can it be measured? For this

study, we are taking the Students final grade for the course as a proxy for a learning outcome. This assumes that our teaching, and assessment practices produce a grade which is a useful measure of learning. The Final Grade is computed by a formula which weights the assessments on the same PAs heavily. Perhaps it would make more sense to use the Final Exam scores (a more limited measure) as the independent variable, the measure of learning outcome.

We are interested in the following:

RQ1: Does giving students the ability to resubmit programming assignments contribute to the learning outcomes.

RQ2: Which particular policies in the policy space have an impact and which ones do not.

## V. Data

In this study there were 76 students overall. The majority were second semester freshmen, intending to be Computer Science majors, but often undeclared. We collected data for each student and each assignment, providing a rich dataset to analyze.

For each student $s$

- Final course score: $CourseGrade_s$:
- For each PA $p$, the initial, resubmission and final scores: $PAInitial_{s,p}, PAResub_{s,p}, PAFinal_{s,p}$
- Average score across all Programming Assignments for this student: $PAmean_s$
- How often they resubmitted (max 11): $ResubCount_s$:
- For each Programming Assignment $p$:
  - Normalized z-score for the PA final score (compared to all other students) $PAZscore_{s,p}$
  - Difference between their initial score and final score (if they resubmitted) $PADelta_{s,p}$

With this rich collection of information, we dive in to see what we can learn.

## VI. Observations

### A. Basic Patterns

*1) How often do students actually resubmit?:* As there were 11 opportunities for resubmission, we can see a fairly robust use of the freedom to resubmit. See Figure 2

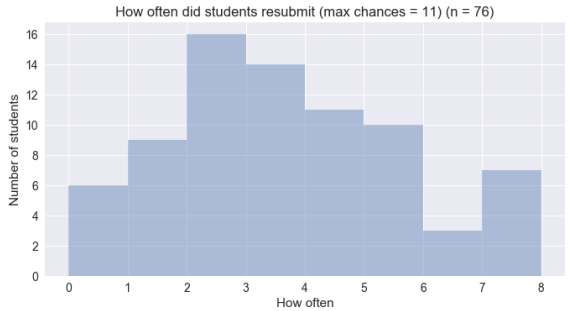


Fig. 2: Distribution of resubmission tries

But why would a student choose not to resubmit? After all it will likely help the score (We found very few cases where the resubmission actually reduced the score for a PA. We believe there are two possible reasons, which we cannot distinguish from our data.

- The student was satisfied with the initial grade, and realized that there was little chance for a significant increase.
- The student just didn't have the time to invest in a resubmission.

This is an important question as we try to divide up results between students who took advantage of the resubmission option and those who did not.

*2) Do scores go up when resubmitting?:* When students do resubmit, do the scores typically go up, and by how much? It is clear from the graph that the scores nearly almost go up, and sometimes by a lot. see. See Figure 3


Fig. 3: Does resubmitted work score higher?

## VII. Clusters

As we showed early on, each PA could have a different degree of difficulty, hinted at by the number of self-reported hours students invested in the solution. Based on this we normalized the individual PA scores based across all students ($PAZscore_{s,p}$)

We gave a lot of thought on how to surface any effects of this policy. We had the following insights:

### A. Early vs. Late

A student's average normalized PA score ($PAZscore_{s,p}$) (across all students and PAs) would show how that student did overall. If we divided up the semester into an early and a late part, we could see whether the student improved between the two periods: take the average, normalized, score for the PAs during the early part of the semester and compare them with the average, normalized score for the PAs during the 'late' part of the semester. For this study we considered the first **five** PAs as the 'early' part of the semester and the remainder as the 'late' part of the semester.

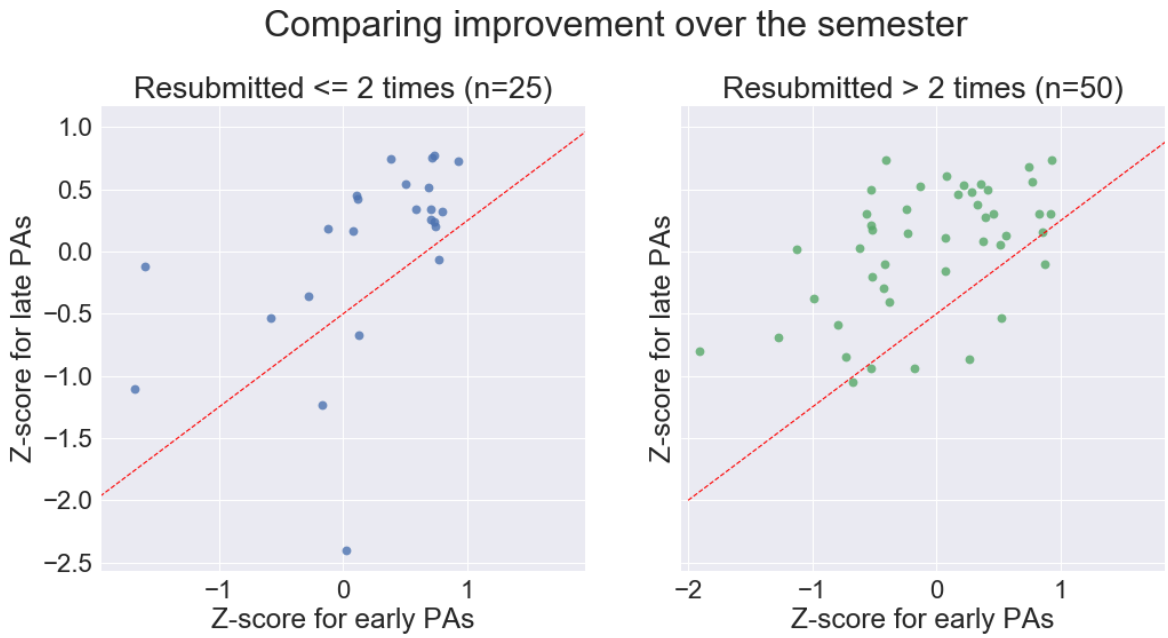


Fig. 4: Can we tell a difference between early and later in the semester?

### *B. Taking advantage of the resubmission policy*

The next question was whether we could partition the students into those who resubmitted and those who did not. It turns out that almost all students resubmitted once. Would it make sense to compare those who rarely submitted to the ones who resubmitted often. If so, what would a reasonable cutoff between the two.

### *C. Excluding the top students*

The final insight was that the effect of the benefit of the resubmission policy would naturally be less pronounced among the students who were not in the top of the class. They would do well in the initial submission and might not be motivated to resubmit, and in any event if they did, this would not have much impact on the score.

### *D. Figure 4: Pulling it all together*

These insights led us to the following way to look at the data. For each PA, over the semester, how did a particular student do compared to the average of the class? And did that particular student get better over the semester?

Figure 4 attempts to illustrate these data. Each student has a dot in these graphs. If the student resubmitted two or fewer times, they appear in the left graph, if they resubmitted 3 or more they would appear in the right graph. In each graph, the x-Axis is the mean score achieved in the first 5 PAs, and the y-axis is the mean score for that student for the last 6 PAs.

So, a student who had the same average score in the early and late PAs would land exactly on the diagonal (45 degree) line. Below the line is a student who got worse over the semester and above the line are students who got better. (The reason there are fewer dots on the left graph is that more students resubmitted 3 or more times.)

In the left subgraph shows students who chose not to resubmit and the right those who resubmitted more often. We have not yet analyzed this statistically so simply visually one may notice that in the chart on the right, more students showed an improvement in performance between the early and late PAs.

## VIII. Further Work

The choice of the course final grade as the ultimate indication of Learning Outcome is easily questionable. Also, the clustering was hand crafted. More analysis might show us a more useful way to cluster the students that would reveal something currently invisible.

We would like to apply other Machine Learning techniques and see whether we discover new patterns in our data. Also we have data which has not been used at all yet, for example the degree of difficulty in each assignment (as indicated by students self-reported hours to completion.) This paper reports our observation and analysis of one instance of one course. In fact, this is the third time this same course has been taught by this same teacher. The first time there were no resubmissions permitted, and the second and third time they were. Bringing these data into our study is an important next area of investigation.

## IX. Acknowledgements

The notion of allowing resubmissions in a computer science course and collecting detailed metrics came to me from Professor Tim Hickey, who also provided comments and feedback on this paper. I also thank Ryan Marcus for his ideas on how to cluster the students,and for his thoughts and feedback on drafts of this paper. Thanks to Prof. Daniel Perlman for his ideas on how to look at the available data.

## REFERENCES


[1] A. M. Holland-Minkley and T. Lombardi. Improving Engagement in Introductory Courses with Homework Resubmission. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*, pages 534–539, 2016.

[2] V. Karavirta, A. Korhonen, and L. Malmi. Different Learners Need Different Resubmission Policies in Automatic Assessment Systems. *Koli Calling '05*, pages 95–102, 2005.

[3] C. Parnin, J. Siegmund, and N. Peitek. On the Nature of Programmer Expertise. *Ppig*, 2017.

[4] H. Sackman, W. J. Erikson, and E. E. Grant. Exploratory Experimental Studies Comparing Online and Offline Programming Performance. *Communi*, 11(1):3–11, 1968.