Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #46981

WIP: Can specification grading with resubmissions improve the quality of student programming?

Dr. Jennifer Mortensen, Worcester Polytechnic Institute

WIP: Can specifications grading with resubmissions improve the quality of student programming?

Abstract

Student programming assignments are a foundation of most computer science courses. Programming assignments allow students to engage with course concepts and develop their programming skills. As a student moves through their computer science courses, it can be easy to focus on whether each program performs its desired function, without thinking about their work's overall quality (documentation, style, and functionality). Specifications grading is one method that attempts to solve this problem by requiring students to meet a set of requirements for each assignment that focuses on the overall quality of the work, which can include programming style, functionality, and documentation. In this paper, we discuss the implementation of specifications grading to motivate students to write high-quality programs and present preliminary results using this grading system in Fall 2024 in a class with 77 students. To assess the impact of the grading system on students and faculty, we analyze students' submission and resubmission patterns. We discuss the impact of resubmission on student procrastination and academic integrity. Finally, we will discuss the problems we encountered with the grading system and our plans to improve the system and implement it in larger enrollment sections of the same course.

Introduction

Programming assignments create the backbone of most computer science (CS) courses. In these assignments, students are asked to apply new concepts and develop a program to solve a predefined problem. The instructor or TA reviews the work, uses a rubric to assess its accuracy or correctness, and slaps on a grade. Often, students read the grade, sometimes view the feedback, and then they move on to their next task. As a result, students focus on the outcome of their work (does it solve the task presented or produce the correct output) but ignore the quality of their work. The traditional grading system lacks an emphasis on program logic, style, and documentation that is necessary for students to grow as programmers and succeed in their future careers.

Alternative grading systems provide ways for instructors to create a feedback loop in their classroom that improves the quality of student work [1]. These systems include standards-based grading, specifications grading and ungrading [2], [3], [4], [5], [6], [7] . Each of these systems attempts to change the meaning of grades and encourage students to produce higher quality work. The application and analysis of alternative grading in CS classrooms is still in its early days and additional work is needed to see how well these grading methods adapt to programming assignments. A recent literature review of specifications grading and contract grading in computer science found only 11 papers evaluating these grading methods in undergraduate CS courses [8]. In this paper, we describe the start of our work evaluating specifications grading in the third course in the computer science sequence at Worcester Polytechnic Institute (WPI). We found that specifications grading improved student work, but that our current implementation led to student procrastination. As we continue to implement this grading in future sections of the course, we will be able to see how changes to the submission requirements reduce procrastination while still requiring students to produce high quality work.

Developing specifications grading for a 2000 level CS course

In this section we describe the context of our CS course at WPI. We then describe the motivation behind our course redesign and the details of the changes we made.

The Course

Our course, CS 2303: Systems Programming Concepts, is the third course in the CS sequence at WPI. The course is taught in a traditional lecture style with 50-minute lectures four days a week for seven weeks, with a total of 28 meetings. Students engage with course material through practice assignments (reading checks, tutorials, and worksheets) before demonstrating their knowledge through projects and exams. The primary assessments are three on-paper exams and six programing projects. In prior iterations of the course, the grading breakdown was 30% exams, 40% projects, and 30% practice assignments. WPI does not award plus and minus grades and students who fail the course earn a "No Record" (NR) which does not impact their GPA. This results in 4 possible final grades in the course: A, B, C, and NR.

A typical section of our course contains 75 to 150 students. To manage the grading workload, programming assignments require students to write programs to process user input and produce output with a specific format. This allows us to use an autograder to automatically run student programs and check the output. All projects are submitted through the Gradescope platform which integrates with Canvas [9]. When a student submits their program, Gradescope compiles it and runs it with several test cases. The output is compared to the expected output and must match to receive autograder points. Students can resubmit to Gradescope until their program meets the autograder requirements. After the deadline, additional manual grading is done to check the program for other rubric items such as documentation and project specific requirements.

The Need for Change

In a traditional grading context, each student project is assessed by a set of criteria to earn a numerical grade. These grades average together with other assignments to produce a student's final grade. In past iterations of this course, we noticed that students could earn an A in the course while frequently missing key components of the project requirement such as never documenting their code. Figure 1 shows the relationship between project grades and the final grade in the course from our Spring 2024 section of our course. A range of project averages is present for each final grade, with project averages as low as 75% earning an A and 62% earning a B. In our opinion, earning an A should require producing professional quality work with well-structured code and high-quality documentation. A traditional grading system makes it difficult to correlate the quality of work to final course grades.

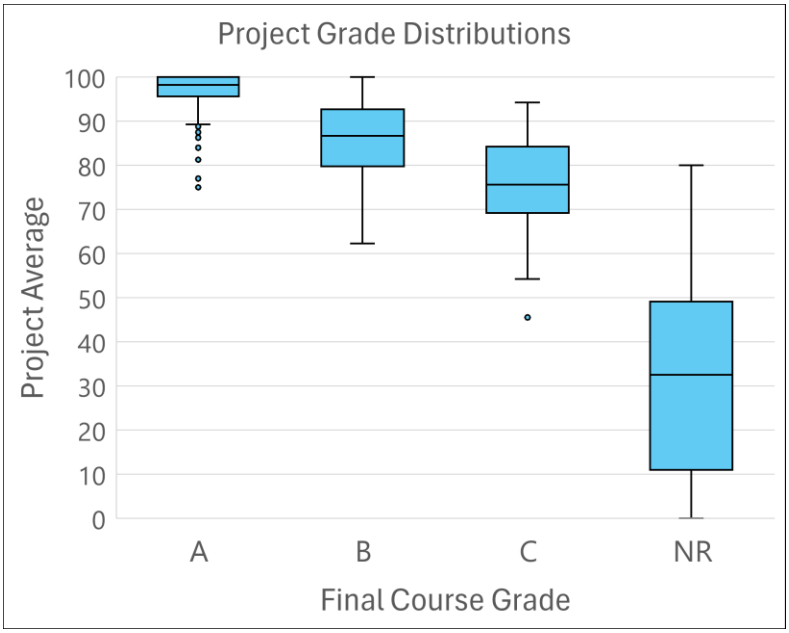

Figure 1: The relationship between final grade and project average in a prior iteration of our course (Spring 2024) using traditional grading. Students have a wide range of project averages for each final course grade earned.

In addition to problems with the quality of student work, the typical format of the course made it difficult to deal with requests for extensions. Late work was penalized by 10 points a day. In our large sections of the course (100+ students), the requests for extensions frequently became unwieldy and required substantial instructor time to review and approve. We also found that the pressure of a late penalty was contributing to academic integrity violations, typically in the form of students submitting a classmate's work as their own. Our hope was that an alternative grading system with flexible submission deadlines would reduce cheating and the need for extensions, while still requiring high quality student work.

Redesigning the Course

We chose to implement specifications grading for student projects. In specifications grading, students are given a detailed list of what a high-quality submission should look like. In our case, this took the form of a rubric requiring well-structured code, quality documentation, consistent style, and project specific requirements. In addition to passing the autograder, each student submission was assessed against each rubric item. Instead of receiving a numeric grade based on the number of rubric items met, student submissions were marked as "Excellent", "Meets Expectations", "Revisions Needed" and "Not Gradeable" (abbreviated E-M-R-N). Submissions that met all criteria earned an "E", those with most criteria earned an "M", and those that missed many earned an "R". The grade of "N" was used for submissions that did not produce the correct output as assessed by the autograder. After each project, students were encouraged to revise and resubmit their work to earn a higher mark. Projects 0, 1, and 2 had a 3-week window to make late submissions or resubmissions. Projects 3, 4, and 5 had shorter windows due to reaching the end of the term (2 weeks, 1 week, and 4 days respectively).

We chose not to modify the exams or practice material in the course and evaluated those assignments with a traditional numeric grading scheme. Students' final grades were based on

their exam scores, total practice material completed, and the number of projects completed with an E or M (Table 1). Students needed to meet all criteria for each letter grade to earn that grade. Failure to meet the requirements for a C resulted in an NR (No Record).

Table 1: Requirements for each letter grade. Earning a given grade requires meeting all of the requirements for that letter grade. Failure to meet the requirements for C will result in an NR.

To earn	Accomplish the following
A	Complete **6 Projects** with at least an M mark, including at least 3 E marks. Earn at least **240 Exam** points. Earn at least **180 Other** points.
B	Complete **5 Projects** with at least an M mark, including at least 2 E marks. Earn at least **200 Exam** points. Earn at least **150 Other** points.
C	Complete **4 Projects** with at least an M mark. Earn at least **170 Exam** points. Earn at least **120 Other** points.

Data collection

The Gradescope submission system retains all student submission and resubmission data. To analyze student submission patterns, we downloaded the submission data for each project and looked at all manually graded submissions for each student.

Results

Completion rate and overall grades

In Fall 2024, 90 students were enrolled in the class. By the end of the term, 3 had formally dropped the class and another 10 did not attend the final exam. Our analysis will focus on the 77 students who completed the course. Most students who completed the course achieved high grades, with 75% of them earning an A or B (Table 2). In addition to the 10 students who did not complete the course, another 11 students received failing grades, mostly due to incomplete assignments and cheating (see section on academic integrity).

Table 2: The breakdown of final grades from A-term of 2024.

	Fall 2024		Prior course (Spring 2024)
Grade	# of Students	% of Students	% of students
A	45	58.4%	70.5%
B	13	16.9%	15.0%
C	8	10.4%	7.7%
NR	11	14.3%	6.8%

Project Submission Data

A key component of our course was the ability to revise and resubmit projects. Students took advantage of this: 75% of students made two or more submissions for Project 0. Over the entire term, 756 total project submissions were made, an average of 10 submissions per student (across 6 projects). Project 0, which was due early in the term, had the most submissions and resubmissions, and the majority of students needed to resubmit their work multiple times before

earning an E or M score (Figure 2). As students became more confident programmers and more comfortable with the expectations of the course, they needed fewer resubmissions. By Project 4, more than 70% of students submitted high quality work, earning an E or M on their first submission.

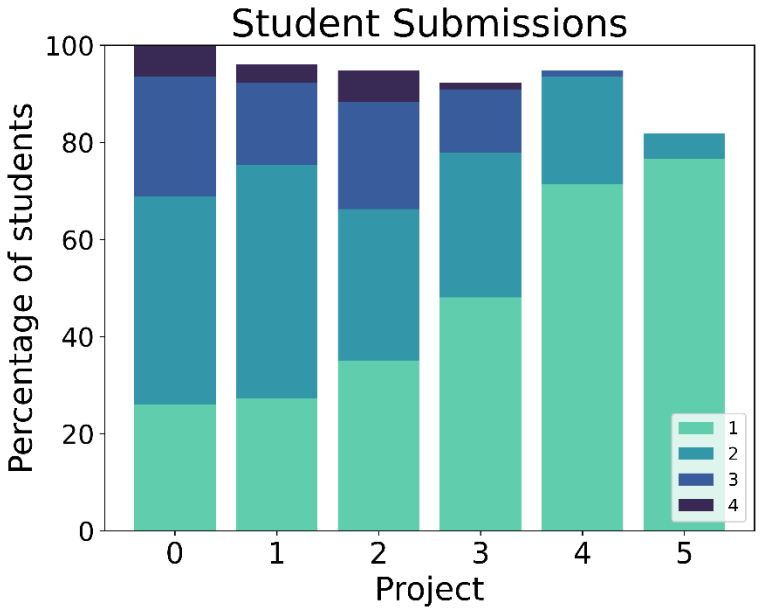

Figure 2: The percentage of students making 1 (green), 2 (teal), 3 (blue) or 4 (dark blue) submissions for each project. 100% of students who completed the course made at least one submission to Project 0. For the remaining projects, some students did not make any submissions.

While students became more proficient as the term progressed, they also took longer to make their submissions. Figure 3 shows the number of submissions made each week of the term across all projects (Total # of submissions / # students). In the ideal world, every student would submit their work on time, (Figure 3, solid black line), and some percentage of students would make resubmissions of their prior work each week. We expected a 50% resubmission rate, giving a total of 115 submissions a week (Figure 3, dotted black line). Instead, we saw much lower submission rates in the first 5 weeks of the term, with a lot of submissions at the end of the term. In weeks 1, 2, and 3, we received less than one submission per student. As we reached the end of the term, students started submitting their late work. On week 7, the last week any work could be submitted, we received an average of three submissions per student.

Academic Integrity

Student submissions were also checked for academic integrity violations using the Gradescope similarity checker. This checker compares all submissions for an assignment against each other and flags any highly similar submissions. All submissions flagged as highly similar were manually checked and the university academic integrity process initiated for all submissions suspected of violating course policies. Five students were investigated for cheating and all five students admitted to either sharing their work with a classmate or submitting another student's work as their own. Prior iterations of the course saw 5-8% of students cheating on assignments, which is in line with the 6% seen in this course.

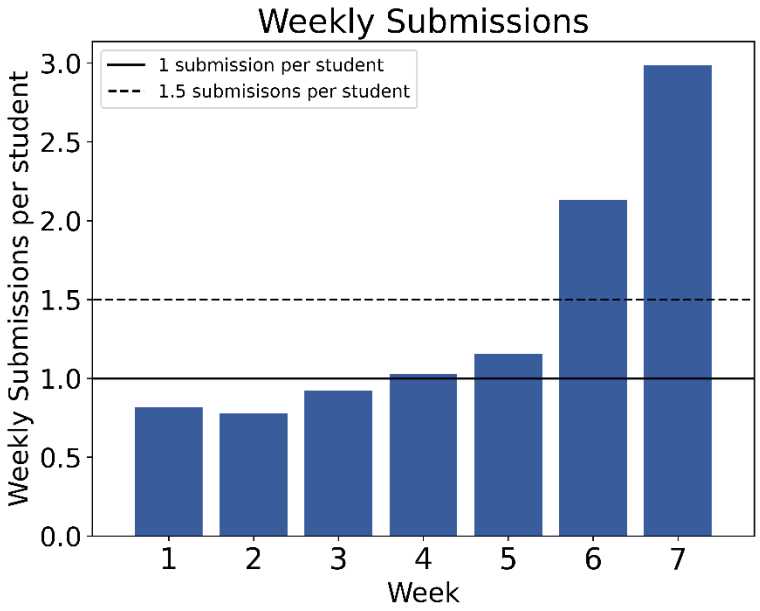

Figure 3: The distribution of student submissions across the 7-week term. The submission rate of 1 submission per week per student and 1.5 submissions per week per student are also shown (solid black and dotted black lines, respectively).

Discussion and lessons learned

Our hope in implementing specifications-based grading was that students would be encouraged to produce higher-quality work and that we would see less student stress and less cheating. We achieved one of these goals: students produced better work as the term progressed. In this section we will discuss how this goal was met and the changes that are needed to meet our other goals.

Producing high-quality work

From the outcomes of this course, we see that specifications grading resulted in a better match between student work and their final grade. In order to earn an A, a student needed to produce high quality work for all of their projects. This lowered the percentage of students who earned As in the course by keeping students with mediocre work from earning an A (Table 2). At the same time, we saw an increase in students failing the course (earning a "No Record" grade).

While we saw an increase in high-quality work, it took students longer than we hoped to get there. Most students needed multiple submissions for the first several projects and many submitted projects days or weeks late. We have identified two key problems in the project submission/resubmission system that were contributing to student procrastination and stress.

1. The initial submission window was too large.
   For the first three projects, students had three weeks to submit late work or resubmissions with no late penalty. For example, if the project was due on September 9th, students could submit it until September 30th. As a result, many students treated September 30th as the due date, starting the project a few days before that date. This left them no time for resubmissions and led to a project submission crunch at the end of the term. This can be

clearly seen in Figure 3 where on average each student made three submissions in the last week of the term. In future iterations of the course, we will test using shorter resubmission windows and requiring an initial submission by the due date.

2. The rubric was too vague.
   In order for students to produce high-quality work, they have to know what high-quality looks like. From student feedback, we learned that the rubric used did not give students enough information about what was expected of them. As a result, they would submit work which they thought was good but would be marked as "Needs Revision." An example of a vague rubric item is shown in Table 3. To improve student buy-in for specifications grading, we need to modify the rubric items to be more specific. This will allow students to feel that the bar being set is achievable and know that their hard work will be rewarded. Having a more specific rubric will also improve the grading process. It is easier for a teaching assistant to determine if an assignment uses global variables, or if it has the correct indentation, than for them to grade the vague concept of "correct style" or "good logic." Having a clear and comprehensive rubric providing students with expectations should produce less confusion for students [10] and improve student views of the grading system.

Table 3: Example of the rubric provided to students and a proposed improved rubric with more details.

Original Rubric
1. Code is clearly written, with consistent style
Improved Rubric
1. Style a. `const` variables are in all caps b. All content in a set of `{ }` is indented c. Variables start with lower case letters d. Program consistently uses camelCase or underscores (but not both) e. Lines of code used for debugging have been removed (no commented out code) 2. Logic a. Each function performs 1 task b. Loops and conditionals have a meaningful purpose c. Global variables are not used

Managing student stress and cheating

Our alternative grading system did not impact student stress and academic integrity in the ways we had hoped. By creating a long submission window, we accidentally encouraged procrastination, resulting in student stress at the end of the term. It has been pointed out by others that students default to a due-date-driven approach of managing their workload, where they only focus on the most urgent tasks [10]. Without external forces (due dates) motivating students to start their work, they procrastinate and do not give themselves enough time to complete their work. Students were also stressed about the unclear expectations for their work due to the vague wording in the rubric

Sadly, allowing for resubmissions did not reduce academic integrity violations. Approximately 6% of our students shared or copied code, similar to prior iterations of the course. One key

difference was that all of the cheating occurred in the last week of the term, instead of being spread out over the entire 7-week period. We suspect this is due to the submission flexibility available throughout the term and the hard deadline at the end of the term. Most of our students who cheat do so because they are overwhelmed by their workload and make the poor choice of copying instead of reaching out for help. While we will continue to look for ways to reduce student cheating, we wonder if it is truly possible to remove the stress that leads to student academic misconduct. Having all of the cheating occur at the end of the term did make the instructor workload slightly easier.

Conclusions and Future Directions

Our first term using specifications grading showed that it can improve student work, but that additional changes are necessary to reduce student stress and cheating. We found that the overall grading load was increased, with a huge spike at the end of the term. Our hope is that changing the resubmission window can spread out the grading workload, keeping it more manageable. We will be testing several submission patterns over the next few terms including shortening the submission window and requiring a submission to be made by the initial due date. We are also planning to perform more analysis on the quality of student work by comparing submissions to projects under specifications grading to those submitted with a traditional grading scheme. Finally, we will be making additional modifications to deal with larger course sizes and will be surveying students to investigate their views of specifications grading.

[1] D. Clark and R. Talbert, *Grading for Growth*. 2023.

[2] E. Dosmar and J. Williams, "Student Reflections on Learning as the Basis for Course Grades," in *2022 ASEE Annual Conference & Exposition Proceedings*, Minneapolis, MN: ASEE Conferences, Aug. 2022, p. 40458. doi: 10.18260/1-2--40458.

[3] J. Garner, P. Denny, and A. Luxton-Reilly, "Mastery Learning in Computer Science Education," in *Proceedings of the Twenty-First Australasian Computing Education Conference*, Sydney NSW Australia: ACM, Jan. 2019, pp. 37–46. doi: 10.1145/3286960.3286965.

[4] K. R. Sanft, B. Drawert, and A. Whitley, "Modified specifications grading in computer science," *J. Comput. Sci. Coll.*, vol. 36, no. 5, pp. 34–46, 2021.

[5] S. Spurlock, "Improving Student Motivation by Ungrading," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, Toronto ON Canada: ACM, Mar. 2023, pp. 631–637. doi: 10.1145/3545945.3569747.

[6] E. Tuson and T. Hickey, "Mastery Learning with Specs Grading for Programming Courses," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, Toronto ON Canada: ACM, Mar. 2023, pp. 1049–1054. doi: 10.1145/3545945.3569853.

[7] R. Weber, "Using Alternative Grading in a Non-Major Algorithms Course," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, Toronto ON Canada: ACM, Mar. 2023, pp. 638–644. doi: 10.1145/3545945.3569765.

[8] B. Harrington, A. Galal, R. Nalluri, F. Nasiha, and A. Vadarevu, "Specifications and Contract Grading in Computer Science Education," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, Portland OR USA: ACM, Mar. 2024, pp. 477–483. doi: 10.1145/3626252.3630929.

[9] A. Singh, S. Karayev, K. Gutowski, and P. Abbeel, “Gradescope: A Fast, Flexible, and Fair System for Scalable Assessment of Handwritten Work,” in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, Cambridge Massachusetts USA: ACM, Apr. 2017, pp. 81–88. doi: 10.1145/3051457.3051466.

[10] P. Gestwicki, “Godot engine and checklist-based specifications,” *J. Comput. Sci. Coll.*, vol. 37, no. 4, pp. 30–40, 2021.