

Internal report: COMP.CS.300 - growing grade-by-grade

Pia Niemelä¹^a, Jenni Hukkanen¹^b, Mikko Nurminen¹^c, Jukka Huhtamäki¹^d

¹Faculty of Information Technology and Communication Sciences, Tampere University, P.O. Box 1001 FI-33014, Tampere, Finland
{pia.niemela, jenni.j.hukkanen, mikko.nurminen, jukka.huhtamaki}@tuni.fi

Keywords: learning management system, next-generation learning environment, assessment and feedback, automatic grading, manual grading, peer-reviews, leaderboards, learning analytics, flipped learning, growth mindset, the theory of formative assessment

Abstract: The enrollment of computer science students continues to increase, with record enrollment numbers in the course "Data Structures and Algorithms" in fall 2022. As a result, teaching methods and systems must evolve to support student progress in the face of scarce teaching resources. This paper examines the shift from manual to peer-reviewed and auto-graded grading processes and investigates students' perceptions of different grading styles. The developed automatic graders utilize data from both the Plussa and GitLab, which serves as the channel for student submissions and provides feedback for formative assessments. The results indicate that peer-reviews are accepted as an exercise for the reviewer, but not as a means to grade the reviewee. Auto-graders are well-received due to their instant feedback, ability to allow for multiple submissions and iterate towards more efficient solutions, which helps foster a growth mindset.

1 INTRODUCTION


The adage "You get what you grade" emphasizes the significance of dependable, impartial, and efficient grading methods. To accomplish this, grading must be designed to provide immediate and ongoing feedback to students, allowing them to make adjustments in real-time and improve their performance in a timely manner. However, this approach must also be able to accommodate ever-increasing student populations, which may seem like a daunting task.


In online learning, students often need to be more autonomous, as scaffolding from the course personnel may be only available in online forums. The learning material is often provided as videos. The videos can be cut in short clips of 15 .. 30 minutes, because cutting the material in shorter portions has proven to increase student engagement in the earlier studies (Saunders et al., 2020; Bergmann and Sams, 2012; Slemmons et al., 2018). The internalization of video content can be effectively assessed through the use of multiple choice questions. The flipped learning approach also suggests the use of "primetime" sessions,


where small groups of students can ask questions to the instructor (Koskinen et al., 2018). Following this, students can apply the learned topics in weekly exercises that involve both theoretical analysis and programming. The programming exercises should aim to minimize the need for additional support from course personnel. To achieve this, clear and well-structured exercise instructions should be provided. Additionally, exercise graders should furnish students with adequate feedback in order to support their learning and development.


In the studied Data structures and Algorithms course (DSA-2022, N=605), the goal legitimized the effort to make all the exercises automatically graded. In previous course implementation, some of the weekly exercises were already automated¹. In this implementation, the effort was made to automate the rest of the exercises, as well as the bigger final course-work assignments and exam.

In this article, we will delve into the advantages and disadvantages of various grading methods, specifically focusing on the comparison between manual and automatic grading. The approach adopted

^a <https://orcid.org/0000-0002-8673-9089>

^b <https://orcid.org/0000-0002-7691-5974>

^c <https://orcid.org/0000-0001-7609-8348>

^d <https://orcid.org/0000-0003-2707-108X>

¹Course personnel in 2023: teachers: Pia Niemelä (responsible teacher), Matti Rintala and Frank Cameron, TAs: Eerik Voimanen, Iiro Sinisalo, Duc Hong, Janne Miilunpalo, Jaakko Rajala and Jenni Hukkanen

is research-based and evidence-based, and the effects of the change are evaluated through monitoring learning outcomes and surveying students about their perceptions of the shift from manual to automatic grading. These findings will be used to inform and adjust future course implementations. To gain insight into how students received the integration of auto-graders, the following research questions will be posed:

1. Which grading styles students prefer the most and why?
2. Which auto-graders were appreciated the most?
3. Which were the students' desired submission counts and what were their consequences?

2 RELATED WORK

After the COVID-19 pandemic, many students continue to request remote teaching in the form of videos and remote exercise sessions, even as in-person teaching resumes. The increased demand for remote teaching and limited resources have led to a shift towards remote teaching as a norm. In this environment, the role of educators and course personnel becomes more implicit, yet they design the course practices and deliver material, exercises, and assignments to best serve their target audience. Learning occurs primarily through the use of learning management systems, which can be enhanced with intelligent tutors. In guiding students, grading is one of the primary means. The purpose of the grading is to promote student's learning and guide the learning process in an appropriate direction, in computer science, improving their coding skills.

To maintain the quality of teaching in an online or hybrid environment, the findings of pedagogical frameworks such as the growth mindset (Dweck and Yeager, 2019), the theory of formative assessment (Black and Wiliam, 2009; Clark, 2012), and flipped learning (Bergmann and Sams, 2014) can inform the use of grading tools while developing the DSA-2022. These frameworks can provide insight into what benefits students in terms of feedback, engagement, and self-directed learning, which can be incorporated into the design and implementation of grading tools. Additionally, the use of formative assessments and active learning strategies, as promoted by flipped learning, can improve the effectiveness of learning.

2.1 Pedagogical frameworks

Pivotal in detecting the growth mindset is a student's attitude toward a challenge. If the challenge feels

thrilling and not intimidating, it speaks for a growth mindset. A growth-mindset student is thrilled by an opportunity of learning something new and difficult. The opposite of the growth mindset is fixed mindset (Haimovitz and Dweck, 2017). Fixed-mindset students are constrained by shame and fear of failure. Ultimately, the fear is induced by the risk of others discovering them being dumb. In a fixed mindset, this would be tragic indeed, because the judgement is final: intelligence and skills are thought to be given and that is why nothing can be done about it. Not so in the growth mindset. It expects students to grow by taking on the challenges, which trains students' flexibility and resilience, and helps them become eventually more skilled. Growth-minded students are also confident about their progress and think there are good chances in achieving the set academic goals, which connects to self-efficacy (Bandura, 1982) Educators should rely on this growth and equip students to regulate their own learning, and learn to learn.

The theory of formative assessment is based on the idea that assessment should be used as a tool for improving both learning and instruction, rather than just evaluating them. Formative assessment takes place throughout the learning process, it also provides feedback not only for students but also for course personnel on what is being learned and what needs to be improved in the course design. E.g., the instructions and exercise feedback are often refactored in anticipation of a smoother completion of exercises. According to Clark (Clark, 2012), there are three aspects of feedback which have the potential to impact meta-cognition and self-efficacy, namely formative, synchronous and external/internal feedback. Clark defines formative as an assessment style that supports self-regulation, synchronous as instant feedback, whereas internal feedback refers to ways persons speak to themselves. Internal feedback generation is typical for self-regulated learners, often triggered by getting external feedback first. (Clark, 2012).

Flipped learning (FL) has grown into a popular approach that involves students watching video lectures before engaging in active learning activities such as problem-solving and discussion. Assessment in FL typically includes formative assessments that take place throughout the course.

In MOOCs, active learning implied by FL may incorporate interactive and gamified exercises and coding. The feedback for these exercises can be categorized as formative, if it is instant and detailed enough, so that students can improve their submissions based on it. The FL approach is believed to better prepare students for the 21st century by giving them more

control over their learning, allowing them to learn at their own pace, still not neglecting the opportunities for active engagement and collaboration (Avery et al., 2018).

2.2 Respective learning tools

Dweck together with her research group have developed a reward system, *"brain points," to encourage the development of growth mindset behaviors by directly incentivizing effort, use of strategy, and incremental progress.* (Dweck and Yeager, 2019; O'Rourke et al., 2014; O'Rourke et al., 2016) Results show that brain points are capable of encouraging the growth mindset by increasing persistence, time spent playing, strategy use, and perseverance, especially in low-performing students. By providing incentives for effort and progress, players are more likely to engage with the game and continue to challenge themselves, thus fostering the growth mindset among players. Students will be encouraged to see challenges as opportunities for growth and development, rather than as a source of frustration.

3 RESEARCH CONTEXT

The studied DSA course provides a comprehensive introduction to data structures and algorithms, from insertion- to merge sort, from data structures sequences and sets and algorithm analysis including the performance implication of data structure selection. Due to COVID-19, DSA course replaced earlier live lectures with video recordings, and on-premises tutoring with online tutoring sessions. Special Q&A sessions were provided for tutoring purposes. In the sessions, teachers answered questions stated by students and went more in detail than in video lectures. Q&A is similar concept to primetime sessions that are utilized in flipped learning (Koskinen et al., 2018), except in DSA-2022 participation was voluntary and not rewarded by points. The removal of points resulted in a remarkable decline in participation compared with previous implementations.

The exercises and assignments demonstrate how well the content was internalized. Students struggling with the exercises could get help in Teams where TAs give hints and scaffold students in solving the problems. However, while doing the exercises students should get a sufficient amount of feedback from the developed auto-graders to be able to fulfill the requirements. The assignments are designed to be completed independently by the students, and successfully completing the assignments is evidence of ad-

equating learning.

Assignment 1 and Assignment 2 are divided into two phases: compulsory and optional. Students have the option to accept the results from the compulsory phase or choose to progress to higher grades. Additionally, students may choose to forgo Assignment 2 entirely, resulting in a maximum grade of 2. These various grade options can be viewed as an implementation of flipped assessment, which grants students more control over their assessments, allowing them to be more autonomous and focus on areas that align with their strengths and interests (Toivola, 2019).

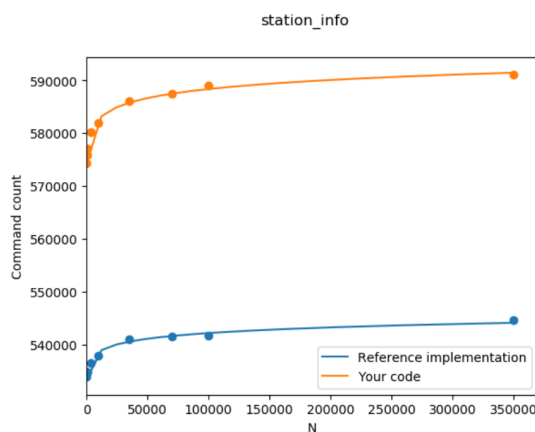
3.1 Tools used: Plussa, openDSA and Gitlab

Plussa is the learning management system that was used in the course. Originally Plussa has been developed in University XYZ (?). Original LMS is a service-oriented architecture that was designed to be easily extendable. This allows for the addition of new services, such as auto-graders, to enhance its functionality (?; ?). openDSA exercises are such enhancements, but basically anything is possible: the docker container with an image of developer's choice will be launched, and the git repository is cloned inside the image and tests will be executed inside this sandbox.

Open Data Structures and Algorithms (openDA) exercises are especially designed to support growth via formative assessment. OpenDSA is an open-source community creative-commons project; a broad community of developers was involved in the effort with the incentive of re-using the materials in their own courses. The University of Virginia has been one of the main promoters; in Finland, Aalto University has been active.

The project has achieved a wide variety of self-study resources for learning data structures and algorithms, such as interactive visualizations, quizzes, and coding exercises to help students learn and practice computer science concepts (Shaffer et al., 2011; Fouh et al., 2012). The visualizations cover different algorithms (Karavirta and Shaffer, 2015; Tilanterä et al., 2020) and runtime behaviors (Sirkiä, 2018), that is, the illustration of call stack and heap behavior while executing an algorithm, such as recursion (e.g., Annotation editor exercise about recursion). Recursion as well as time complexity are identified as one of the threshold concepts in computer science (Zander et al., 2008), and visualizations are an apt tool for lowering the threshold (Shaffer and Rosson, 2013).

openDSA comprehension aids are good for novices, but to advance students' craftsmanship in software engineering the courses need no toys but



(a) Assignment1

next_stations_from

- N=10000 ✓
- N=35000 ✓
- N=70000 ✓
- N=100000 ✓

(b) Assignment2

Figure 1: Perftests: resource-consuming and resource-savvy versions

real tools for improving the efficiency and quality of code. Submission through GitLab repos and having real unit and integration tests would better future-proof their development as becoming coders (Haarinen and Lehtinen, 2015).

In addition to Plussa and openDSA, Gitlab is a central tool starting from the middle of the course. Besides functioning as a normal version control system, Gitlab is the means to get instructions pulled from the course upstream and, on the other hand, submit code for grading. The course upstream is a Gitlab repository for pulling only. Course personnel maintain the upstream, new instructions and possible file skeletons are released at the beginning of each exercise round. To complete the course, students had to pass weekly exercises, a coursework assignment, and an exam. The manual assessment of assignments is the most resource-consuming task, thus its automation was the first priority.

3.2 Auto-graders for assignment

In 2022, the automation of the course was leveled up by introducing multiple auto-graders to check various aspects of students' code.

Fig.2 illustrates auto-graders in action. First, students pull instructions from the course-upstream repository. They then commit their code to their own repository, and finally, submit the Gitlab URL in Plussa system, which is responsible for submission and grading. Plussa system is divided into two parts: the Plussa front-end and the MOOC grader. The system launches temporary Docker containers that are only started for performing the grading. The grader clones the student's git repository and executes the grading as instructed in a shell script. Examples of graders include perftest, unit, integration and Val-

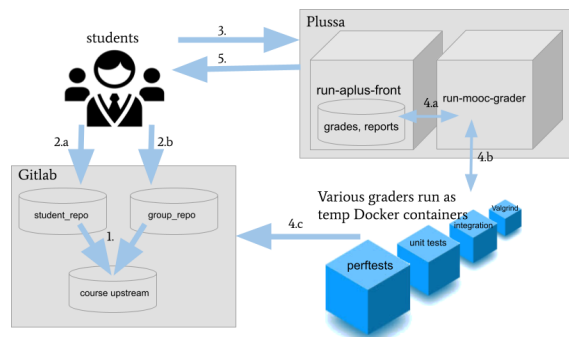


Figure 2: Plussa and Gitlab co-operate during grading

grind graders. Most of the tests are also given to students, and the respective Plussa graders run the same tests. By running the tests locally, students receive the same feedback as given by the Plussa graders, which decreases the number of needed submissions. This also gives students an idea of how their work will be graded in the LMS. After grading, the points are returned to the Plussa front-end, which stores the grades.

The primary goal of Data Structures and Algorithms (DSA) is to learn how to write efficient code. This goal is emphasized by setting more emphasis on performance testing, with the measurement of the time that program takes to run, i.e., its performance. However, measuring performance using a stopwatch can be affected by other processes running on the computer at the same time, thus competing for the same resources. To overcome this issue, an instruction counter is used instead, which measures the number of instructions that are executed by the program. It is not susceptible to interference from other processes. However, the instruction counter requires certain conditions to be met, such as having perf events

open on the server. To meet these requirements, a dedicated ESXi server with an instruction counter was setup for this course implementation.

In DSA-2022, performance tests were conducted by gradually increasing the amount of data (e.g., $N=10$, $N=100$, $N=1000$, ..). The instruction counts were recorded, and then a curve was fitted using Python to estimate the average complexity of the algorithm. The results were illustrated as a graph, showing the average time complexity as a function of the number of data points, see Fig. 1. To ensure accurate results, the input data used for curve fitting should have minimal noise and the tests should be repeated multiple times to average out the effects of randomized data or other sources of noise.

3.3 Method and data collection

The DSA course is developed on a yearly basis. The method used is a design-based research (DBR) approach, which involves cyclical development with reflective redesign phases (Cobb et al., 2003; Reimann, 2011; Ericson et al., 2016). In DSA-2022, this approach is guided by the theory of formative assessment (Black and Wiliam, 2009; Clark, 2012) and involves combining educational solutions with empirical interventions and proof. The DBR approach includes four stages: design, development, enactment, and analysis (Anderson and Shattuck, 2012; Wang and Hannafin, 2005; Ørngreen, 2015). The cycle represents a course term and the retrospective analysis is used to inform the design of the next implementation.

Students' feedback was collected in two phases, before Assignment 1, and after Assignment 2. In addition, we utilize the grader feedback and logs in scanning the learning process. The redesign of the course is done based on the results, and adjustments are made, where feasible.

4 RESULTS

We collected students' views with two grading related questionnaires during DSA-2022. The first, pre-questionnaire, was carried on in the middle of the course, in module 8 ($N=360$) before Assignment 1. The second questionnaire was executed as a post questionnaire in module 14, after Assignment 2 ($N=274$). 240 students answered both questionnaires.

The questionnaires contained both Likert-scale and open-ended questions. First, we scan through the Likert-scale questions about the preferred grading styles. Likert scales are divided in seven levels from 'Strongly disagree' (0) to 'Strongly agree' (7), and in-

terpret the results by selecting enlightening quotations from students' responses.

4.1 Preferred grading styles

When transferring from resource-intensive manual grading to auto-grading, the main dilemma is how to keep up the good quality of feedback. During this course, alternative ways of giving feedback were experimented, such as auto-graders, peer-reviews, and comparisons with others. Learning analytics and self-reflection were also mentioned, and students were asked to rate these grading methods, yet they were omitted during this implementation.

In Fig. 3, the automatic grading gets upvoted the most in both phases, followed by learning analytics and manual grading, but as we can see, in the latter Fig. Assignment 2, the difference between automatic and manual grading is significantly less (in pre-test, 1.98, in the post 1.2, see Table 1). The reasons for automatic grading dropping in are the problems with perftest in Assignment 1, yet the simplified (thus faster) perftests in Assignment 2 managed to compensate for the felt discomfort.

4.1.1 Auto grading

Auto-grading was by far the most appreciated grading method. In their comments, students praised the speed of the graders and the immediacy of the feedback. In addition, students greeted with joy the possibility to return their solution "unpolished" multiple times, such as the following respondent, *-Also auto graders don't (hopefully) think you are a complete idiot if you try to submit something that is not yet well polished.* The uniformity and fairness of the evaluation was emphasized especially in comparison with peer-reviews that were not trusted that much. To a minor extent, manual grading was attached with trust issues as well, such as: *- Personal written feedback would've been very nice, but emphasis on the word personal.* and *- There could also be some unknown bias causing certain students to receive better/worse grades than they deserve.*

Auto-graders have advantages in terms of transparency and continuous evaluation (characterized by formative assessment) allowing for intermediate inspections and communication with the grader. However, there were issues with Assignment 1, specifically with perftests, which compromised fairness and caused timeouts due to the slowness of visualizing and estimating asymptotic efficiency. These problems were addressed in Assignment 2 by simplifying the perftests, resulting in a smoother deadline. Refactoring of Assignment 1 perftests is the top priority.

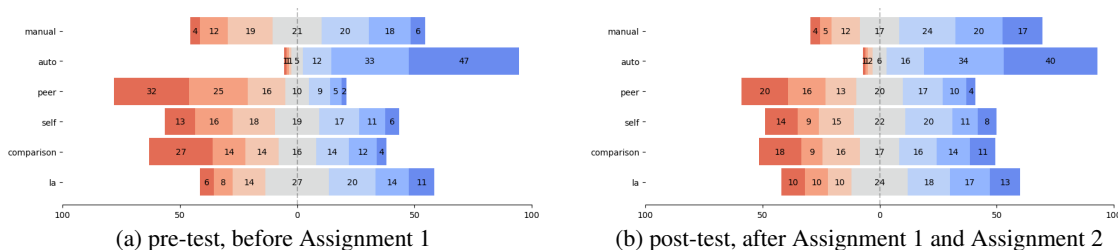


Figure 3: Preferred grading style

	average pre	average post	diff of post and pre	std of the diff	% of changed opinions
manual	4.14	4.8	0.66	1.66	76.25
auto	6.12	5.96	-0.16	1.31	54.17
peer-review	2.54	3.45	0.91	1.73	73.33
self-reflection	3.52	3.89	0.37	1.91	72.50
comparison	3.45	3.86	0.41	1.83	70.83
la	4.17	4.29	0.12	1.96	75.00

Table 1: Summary of statistics for the preferred grading style for the 240 students who answered to both questionnaires

4.1.2 Manual grading

Many students prefer manual grading because of its depth and being personalized. A student pointed out the need for constructive feedback, e.g. hints for better readability and efficiency, and coding conventions. (Yet checking of coding conventions can be achieved with linting tools.) However, sometimes the feedback can degrade as dealing with irrelevances, such as highlighted in the following response, *-Manual grading has on previous courses put enormous weight on nitpicking, which was non-existent now.* Students were also concerned about putting their effort to the exercises in vain, *If a student cannot do code that passes tests but still has SOME code and has used many hours for the work, the grade cannot be 0. There needs to be some manual grading for that pass/fail situation. Like, you could try just to pass (grade 1) OR get a better grade with autograders.* For example, Valgrind grader represents such a pass/failure grader: it did not allow any memory leaks for pass, causing 'passing panic' among students. While universities must maintain the quality of graduates, the amount of work should not automatically compensate for lack of skill. It seems that students rely on humans being more empathetic than machines in this sense.

Multiple responses highlighted the size of the course as a constraint, where manual grading was identified as the method that consumed the most teaching resources, thus it was seen as unrealistic. Additionally, the slowness and an option of returning work only once were seen as downsides of it. How-

ever, in regard to fairness, manual grading received fewer negative comments than automatic grading.

4.1.3 Peer-reviews

In the pre-questionnaire, students were very suspicious about the functionality of peer-reviews. Ignoring the initial dismay, peer-reviews were introduced in the last module of the course, and allegedly the strongest objection has softened after the peer-review period. The primary reason for softening was the fact that peer-reviews did not directly contribute to the grade of reviewee. Also seeing others' solutions was enlightening. However, each review is not equally worthy, *-I think peer-reviewing is a good idea, however I would have wanted to see a solution that was more performant than mine, or alternatively get feedback from an author of such a solution.* and *-Peer reviews depend on the reviewer. Currently it's a pure lottery. If a peer really reads the submission and knows how to give meaningful feedback, peer-reviews are good.* and *I dislike peer reviews as the reviewing is not done by a person that has a clear understanding of the grading in the course and I feel like I am not being graded as fairly as I could have.*

Students desire feedback from someone whose domain knowledge is equal or better than their own. To address this, the simplest solution is to increase the number of reviews, as this increases the chances of receiving feedback from a knowledgeable reviewer. However, in some courses increasing the number of feedback items a student has to give has led to a per-

ceived decrease in the quality of the feedback. Peer-reviews are inherently less equal in quality when compared to automatic grading, as they are subjective. Students should be given incentives to provide high quality feedback to their peers. Additionally the grading of the outcome of the peer-review process should be considered in order to ensure the quality of the review. The quality checking of the reviews calls for innovative ideas. Extending the peer-review process to include students assessing the feedback they have received could be a relatively straight-forward way to improve the process.

Students find it difficult to comment on work if they are still beginners or there is a gap in skill levels, but with proper instructions, the task becomes easier. Thus, the conclusion is to consider increasing the number of peer-reviews and put effort into providing clear and useful instructions to aid the process.

4.1.4 Comparisons

In DSA-2022, the comparison meant perftest top-10 leaderboard. The leaderboard listed students' initials hinting the name but not revealing it fully. Students listed in the leaderboard may take it as an honor. Of course, being omitted from the board may be discouraging, or felt unequal: *-Comparison with other students is good but if there is only the 10 best, hundreds of people get left out and don't see how they compare. so i think that everyone should be displayed on the leaderboards.* Leaderboard has its problems either way, whether with only selected people or all included, yet the ethical problems would peak in publishing "bottom boards" .

In students' responses, the leaderboard induced a lot of negative feedback. The following collection itemizes the reasons for strong objection:

Making this a competition lets only the top students flex their coding muscles when those who need actual help might feel very stupid.

Seeing other peoples progress puts too much competitive pressure on students. The reality is that sometimes we must prioritize our courses and work in order to stay sane and still learn as much as possible. Seeing other people's progress to that function only wrecks havoc.

Comparing yourself to others can be really harmful to your self-esteem and many students already struggle with mental health and imposter syndrome.

In previous comments, competition was considered having a negative effect on one's motivation. Opposite opinions exist, too: *I'm very competitive so getting leaderboards is fun and increases my motivation substantially. After reading the internal review I almost wanted to go back to working on prg1 just so I*

could see if I could get my perfs down to the best ones :D, taking others' progress as a positive challenge exemplifies the growth mindset.

The information of the leaderboard can be also found as an additional means of guidance: *If we were shown some leaderboards during the development, it would have given us some indication that we were on the right path with the project.* Leaderboards and comparative statistical analysis can be provided for students as extra means to follow their progress.

4.1.5 Learning analytics

In larger courses, such as DSA-2022, all additional and summarizing tutoring would be welcome. The attempt is to use statistical information at the benefit of students, yet the exact means are to be found. It is also good to be aware of the delicacy of the issue and focus on discrete handling of the data. Most of all, students wish encouragement and scaffolding with hard exercises, and desire learning analytics for personalized exercises: *I would be open to learning analytics, if they are user-friendly and give you gentle suggestions and encouragement when you do well.* However, at the same time they marvel how personalizing will affect the grading, for example: *I don't know how this would be implemented in practice; would everyone be able to get as many points if different exercises are suggested, would everyone be able to complete the same exercises even if they aren't suggested? Making grade predictions could be useful.*

4.1.6 Fairness aspects

"*I have been graded fairly*" shows that most students can confirm the claim, yet a group of students strongly disagree with it. Responses to questions "*I deserve a better grade than I am getting*" and "*I am getting a better grade than I deserve*" primarily are left-biased, that is, heavily disagreed, yet the latter even more heavily. In accordance with human nature, students would easily accept a better grade, not vice versa.

The claim "*Satisfied with the feedback*" indicates that the feedback provided to the students is well-received, even though there may be room for improvement. The claim "*I would have preferred personal written feedback from the course staff*" is also agreed, but not as much as the previous question, this means that while students are generally satisfied with the feedback they got, they would have preferred to have personal written feedback from the course staff.

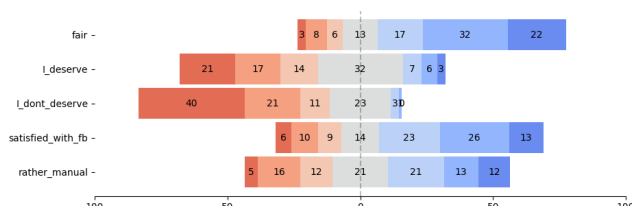


Figure 4: Fairness of grading

4.2 Most useful graders

As seen in Fig. 5, unit and integration tests were considered to be the most useful form of testing, with visualizations provided by performance tests also being highly valued. However, the use of Valgrind as a grading tool was met with mixed reactions due to its strict nature as a gatekeeper that must be passed in order to receive a grade. The main issue with Valgrind is that it does not allow for any memory leaks or errors and thus some participants found it to be too decisive.

However, it is questionable to remove graders that check the correctness, where the removal would eventually lower the quality of code. To address the difficulties with Valgrind, providing better resources and guides for effectively using the tool is crucial. This could include incorporating Valgrind checks when learning about invalidating pointers and provide a guide for tackling the most common memory leaks and Valgrind errors. This way, developers can learn how to use the tool in a more effective and efficient manner and avoid common pitfalls.

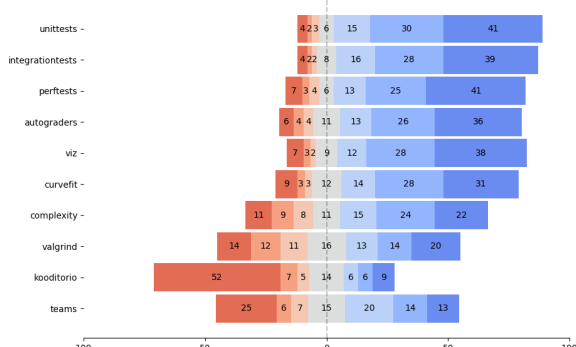


Figure 5: Usefulness of the auto-graders

4.3 Submission count

The submission count, which refers to the number of possible submissions to auto-graders, was initially high at 150 but was later, in Assignment2, decreased to 20 with the ultimate goal being 3.

The desired submission count was asked in the second survey, and the results are shown in Fig. 6.

Responses that suggest submission count of 50 or less total three quarters. The weighted average is still far from the intended goal of ten. Fig. 7 shows the histogram of the submission counts that students used for one particular grader in Assignment 1. The grader is perftest and it was the most heavily used of all the graders. It is still to be noted that 94% of students submitted at most 20 times, even if there are a few students that needed more than 50. In Assignment1, the unit, integration and performance test codes were released in the middle of the submission period; whereas in Assignment 2, the test codes were readily available right from the start. The test codes allowed students to test their code locally, and for Assignment 2 the submission counts were much smaller than in Assignment 1.

This suggests that the majority of students are able to test their codes, if test codes are released. However, students desired to have higher submission counts in auto-graders than they actually needed. This may stem from the need to ensure the possibility to test with the grader, if the test codes are not made available or they behave differently in local testing versus in the learning management system. In any case, students must be encouraged to test their codes by themselves rather than testing against ready-made tests.

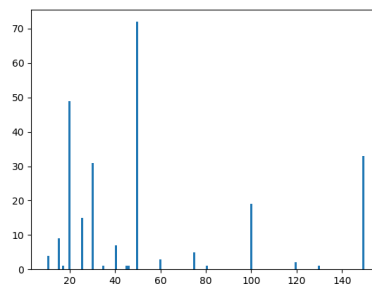


Figure 6: Desired submission counts

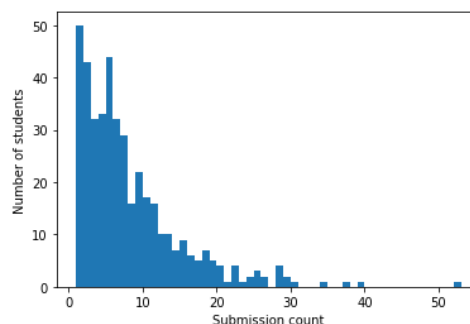


Figure 7: The histogram of the submission counts to the perftest grader in Assignment 1

Had the students been graded manually, this high

number of submissions and this much feedback were far beyond reach. In our university, a few previous courses have been transferred from manual to automatic grading (Niemelä and Nurminen, 2020; Nurminen et al., 2021). The previous experiences confirm the results of this study: if students are provided with an option of gradual improvements of their code, students fully utilize the option, and learn gradually by iterating their code. Another, self-evident but not any less important consequence is saving the resources of course personnel. In the course of this size (N=605), auto-grading is simply a necessity. High submission counts, versatile formative feedback of various graders combined with discussion forums with Q&A, where saved course resources can be spent and humans give constructive feedback, make the virtue out of necessity. However, more effort should be put to foster social cohesion and collaboration among the courses. Post-COVID students of our university have increasingly moved into remote studying. To bring them back to campus and to get them to work in teams is a brand new challenge.

5 CONCLUSIONS

1. Which grading styles students prefer the most and less and why?

Auto-grading was by far the most appreciated grading method, yet manual grading had also strong support but felt unrealistic in the course of this size. It is appreciated because of its depth and being individual. Students also pointed out the need for constructive feedback, and leniency. Peer-review collected the most negative feedback, but views about peer-reviewing got more neutral after students had more experience with DSA-2022 implementation of the peer-reviews: students found it beneficial to see alternative solutions, but others had difficulty providing useful feedback and felt frustrated when the reviewed solution was better than theirs. The feedback from ill-motivated peer-reviewers was considered useless.

2. Which auto-graders were appreciated the most?

Unit-tests, integration tests and perftest, whereas Valgrind grader raised mixed opinions. Perftests, in particular with visualizations, help in reaching the learning goals of the course, efficient code being the very essence of it. The current implementation must though be revised, the perftest grader being much too slow, thus causing congestion. The revised version must function reliably,

despite other processes consuming CPU. Preferably, a browser should draw the curves of asymptotic efficiencies and do curve fitting, instead of a server, which would save its resources and improve the user experience.

3. Which were the students' desired submission counts and what were their consequences?

Students wish for quite high submission counts; the mode was 50 and a remarkable section voted for 150 as well. In reality, they should not need that much, and fortunately the majority did not use that many submissions. Instead of submitting to Plussa, students were instructed to test locally, which is faster and, moreover, aligned with the traversal learning goals of the CS faculty (test-your-code). 150 as the initial submission count of Assignment1 was clearly a wrong signal: instead of leaning on Plussa, students should put more effort on testing their code locally, and preferably they would complete the given test set by writing their own tests, if something is missing.

6 FURTHER STUDIES

The data collected by Plussa and GitLab is extensive and could be used for learning analytics. The analysis should be made accessible to both teachers and students, with the potential for students to compare their performance to others, although this might create unnecessary competition, yet a safer approach would be for students to compare their performance to their own earlier performance. Plussa graders currently check code quality and conventions, but an additional "self-reflection grader" would be useful in helping students to improve their understanding of their strengths and weaknesses, preferably by providing suggestions for exercises to fill in any gaps, and showing the path to growth.

An interesting area of research would be investigating the most effective way to combine automatic grading and the support provided by course personnel. While automatic grading has been shown to be effective and efficient, many students have expressed a need for support from course personnel. The time saved with automatic grading could be used to provide this support and improve teacher-student communication.

Currently Teams channels have been utilized in student-peer and student-teacher interactions, increasingly so during the current COVID-19 pandemic. The interaction strategy on the course primarily focuses on selecting the appropriate tools, such as Teams channels or emails, but the actual usage of

these tools, what is communicated, using which tool and by whom is often decided in an ad-hoc manner. A more structured approach would make communication more predictable and organized. Also the potential to increase social cohesion and engage students could be better utilized. Incorporating team-based activities and assignments can also provide opportunities for students to practice collaboration and communication skills that will be valuable in their future work.

REFERENCES

- Anderson, T. and Shattuck, J. (2012). Design-based research: A decade of progress in education research? *Educational researcher*, 41(1):16–25.
- Avery, K., Huggan, C., and Preston, J. P. (2018). The flipped classroom: High school student engagement through 21st century learning. *in education*, 24(1):4–21.
- Bandura, A. (1982). Self-efficacy mechanism in human agency. *American psychologist*, 37(2):122.
- Bergmann, J. and Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. International Society for Technology in Education.
- Bergmann, J. and Sams, A. (2014). *Flipped learning: Gateway to student engagement*. International Society for Technology in Education.
- Black, P. and Wiliam, D. (2009). Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education)*, 21(1):5–31.
- Clark, I. (2012). Formative assessment: Assessment is for self-regulated learning. *Educational Psychology Review*, 24(2):205–249.
- Cobb, P., Confrey, J., diSessa, A., Lehrer, R., and Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, 32(1):9–13.
- Dweck, C. S. and Yeager, D. S. (2019). Mindsets: A view from two eras. *Perspectives on Psychological science*, 14(3):481–496.
- Ericson, B. J., Rogers, K., Parker, M., Morrison, B., and Guzdial, M. (2016). Identifying Design Principles for CS Teacher Ebooks Through Design-Based Research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16*, New York, NY, USA. ACM.
- Fouh, E., Sun, M., and Shaffer, C. (2012). Opensa: A creative commons active-ebook. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 721–721.
- Haaranen, L. and Lehtinen, T. (2015). Teaching git on the side: Version control system as a course platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, pages 87–92.
- Haimovitz, K. and Dweck, C. S. (2017). The origins of children’s growth and fixed mindsets: New research and a new proposal. *Child development*, 88(6):1849–1859.
- Karavirta, V. and Shaffer, C. A. (2015). Creating engaging online learning material with the JSAV javascript algorithm visualization library. *IEEE Transactions on Learning Technologies*, 9(2):171–183.
- Koskinen, P., Lämsä, J., Maunuksela, J., Hämäläinen, R., and Viiri, J. (2018). Primetime learning: collaborative and technology-enhanced studying with genuine teacher presence. *International journal of STEM education*, 5(1):20.
- Niemelä, P. and Nurminen, M. (2020). Rate your mate for food for thought: Elsewhere use a grader. In *Proceedings of the 12th International Conference on Computer Supported Education - Volume 2: CSEDU*, pages 422–429. INSTICC, SciTePress.
- Nurminen, M., Niemelä, P., and Järvinen, H.-M. (2021). Having it all: auto-graders reduce workload yet increase the quantity and quality of feedback. In *SEFI Annual Conference: Blended Learning in Engineering Education: challenging, enlightening—and lasting*, pages 385–393.
- Ørngreen, R. (2015). Reflections on design-based research. In *Human Work Interaction Design. Work Analysis and Interaction Design Methods for Pervasive and Smart Workplaces*, pages 20–38. Springer.
- O’Rourke, E., Haimovitz, K., Ballweber, C., Dweck, C., and Popović, Z. (2014). Brain points: A growth mindset incentive structure boosts persistence in an educational game. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3339–3348.
- O’Rourke, E., Peach, E., Dweck, C. S., and Popovic, Z. (2016). Brain points: A deeper look at a growth mindset incentive structure for an educational game. In Haywood, J., Aleven, V., Kay, J., and Roll, I., editors, *Proceedings of the Third ACM Conference on Learning @ Scale, L@S 2016, Edinburgh, Scotland, UK, April 25 - 26, 2016*, pages 41–50. ACM.
- Reimann, P. (2011). *Design-based research*, pages 37–50. Methodological choice and design. Springer.
- Saunders, F., Gellen, S., Stannard, J., McAllister-Gibson, C., Simmons, L., and Gibson, A. (2020). Educating the Netflix Generation: Evaluating the impact of teaching videos across a Science and Engineering Faculty.
- Shaffer, C. A., Karavirta, V., Korhonen, A., and Naps, T. L. (2011). OpenDSA: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on computing education research*, pages 112–117.
- Shaffer, S. C. and Rosson, M. B. (2013). Increasing student success by modifying course delivery based on student submission data. *Inroads*, 4(4):81–86.
- Sirkiä, T. (2018). Jsvee & kelmu: Creating and tailoring program animations for computing education. *Journal of Software: Evolution and Process*, 30(2):e1924.
- Slemmons, K., Anyanwu, K., Hames, J., Grabski, D., Ml-sna, J., Simkins, E., and Cook, P. (2018). The impact of video length on learning in a middle-level

- flipped science setting: Implications for diversity inclusion. *Journal of Science Education and Technology*, 27(5):469–479.
- Tilanterä, A. et al. (2020). Towards automatic advice in visual algorithm simulation.
- Toivola, M. (2019). Käänteinen arviointi. *Helsinki: Edita*.
- Wang, F. and Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research and Development*, 53(4):5–23.
- Zander, C., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., and Sanders, K. (2008). Threshold concepts in computer science: A multinational empirical investigation. In *Threshold concepts within the disciplines*, pages 105–118. Brill.