A Comparison of Immediate and Scheduled Feedback in Introductory Programming Projects

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Paul Denny
The University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Jacqueline Whalley
Auckland University of Technology
Auckland, New Zealand
jacqueline.whalley@aut.ac.nz

ABSTRACT

How students are assessed has a powerful effect on their strategies for studying and their learning. When designing assessments, instructors should consider how different approaches for providing feedback to students could encourage positive learning behaviours. One such design is the use of interim deadlines that enable students to receive and respond to feedback. This is used to encourage students to start early and thus reduce the negative effects of procrastination. If multiple submissions are allowed, penalty schemes can be included to encourage students to reflect deeply on the feedback they receive, rather than developing an over-reliance on autograders. In this work we describe two approaches to feedback used over two consecutive semesters for a final project in a large introductory programming course. In both semesters, the complexity and structure of the final project was similar and students received identical instruction. In the first instance of the course students could submit their work prior to two scheduled interim deadlines, after which they would receive automated feedback, before meeting a final third deadline. In the second instance, students received automated feedback immediately upon submission but with increasing penalties to discourage excessive submissions. In both cases, the ability to receive automated feedback – both scheduled and immediate – was designed to encourage early participation with the project. Under the two feedback schemes, we observed different patterns of behaviour – particularly for the lower performing students. We explore the benefits and drawbacks of the two schemes and consider implications for future project grading.

CCS CONCEPTS

• **Social and professional topics** → *Computing education.*

KEYWORDS

feedback, assessment, grading, deadlines, interim deadlines, automated feedback, procrastination, at-risk students

ACM Reference Format:
Juho Leinonen, Paul Denny, and Jacqueline Whalley. 2022. A Comparison of Immediate and Scheduled Feedback in Introductory Programming Projects. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499372

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA

ACM ISBN 978-1-4503-9070-5/22/03…$15.00
https://doi.org/10.1145/3478431.3499372

1 INTRODUCTION

Giving students feedback in introductory programming courses is important for their learning [27]. In traditional introductory programming courses, students typically receive feedback for their project work sometime after the deadline for the project has passed. Such feedback is often produced manually by markers or teaching assistants, and can take considerable time and effort to produce. When feedback is only received by students after the due date, and no revisions are allowed, it limits their opportunity to learn since there is no incentive to improve solutions based on the feedback.

One common feedback method utilized in contemporary programming courses is to give students automated feedback and marks generated by an autograder [1, 22]. The adoption of autograders at least in some part has been pragmatically motivated by large classes and growing enrolments in computing degrees and courses. Assessment structures that use autograders have the advantage that they allow for more than one submission and allow students to receive immediate feedback. The feedback provided can then be used by students to change and improve their solutions. Such assessment models provide genuine feedback and follow the ideology of assessment for learning [41]. How and when feedback is given is known to affect student behaviour. Prior work has found that students are more likely to submit work close to deadlines [14, 29]. Arguably, this is an undesirable behaviour that negatively affects student performance.

When choosing how students are graded and how feedback is given, one aspect instructors often consider is adopting mechanisms that influence student behaviours in a positive way. For example, as some students procrastinate on assignments and their performance is negatively affected as a result [21], some studies have focused on influencing students' time management with explicit interventions such as reminder emails [15]. Another option is to include fixed interim deadlines where students receive early feedback on their programs and can adjust their solutions before the assessment's final deadline [14]. Instructors can also choose to let students receive immediate feedback based on autograders [27]. However, immediate unlimited feedback from autograders can result in over-reliance on the feedback [3]. Some solutions that have been proposed in order to reduce reliance on automated feedback are: submission limits [1], timers [30], or penalties for excessive submissions [3]. Nevertheless, there is no consensus on how feedback should be given to students in a way that maximises the benefits while avoiding problems related to over-reliance.

In this experience report, we present results from two iterations of an introductory programming course where two different approaches to automated feedback were trialed. In the first approach, there were two interim optional deadlines at which point students would receive automated feedback on work they had submitted. In the second approach, students could access immediate feedback with increasing penalties being applied for multiple submissions to curb over-reliance on the autograder. Our primary goal in this work is to understand how different schemes for giving feedback affect students' behaviour, and to especially focus on how at-risk students are affected, since prior work has found that such effects might be more pronounced for lower performing students [24].

2 RELATED WORK

Many traditional computer programming feedback mechanisms limit the opportunity for students to learn from the feedback. Providing learners with useful, actionable feedback is considered critical to learning and is known to affect engagement and performance [35, p. 1930]. Moreover, this feedback must be timely and detailed in order for it to have any real educational value [35]. Consequently, it is not sufficient to consider only the nature of the feedback provided in response to an assignment – as educators we must also consider how the feedback can be used by students [34].

2.1 Feedback mechanisms

The first time a novice encounters feedback on their programs is usually when dealing with cryptic feedback, from an interpreter or compiler, on syntax [12]. As a result there is a growing body of research into the provision of improved and easy to understand error messages [4, 13]. Another type of immediate and automatically generated feedback that novices may encounter is in the form of output from unit tests. This type of feedback has been shown to help novice programmers write specification compliant and functionally correct code [8, 9]. However other researchers have reported that while novices find this type of feedback helpful for improving their code they also find unit test output difficult to interpret [40]. Some researchers have examined software metrics [6, 28] as an alternative feedback mechanism but found that novices struggle to interpret metric-based feedback [28].

Automated feedback systems, or autograders [11, 16, 17, 32], are reported to be convenient for students and provide useful process data for instructors [23, 36]. Students generally view automated feedback systems positively [19] and the benefits of such systems are well documented [38]. While autograders are acknowledged to be a valuable tool for managing assessments in large classes, there are downsides to their use. In a recent paper "STOP THE (AUTOGRADER) INSANITY", Baniassad et al. highlight the issue of student over-reliance on autograders [3]. Students tend to use such systems as a crutch frequently submitting and using the automated feedback to guide their work rather thinking for themselves: "students tweak and submit as often as they can and predominantly rely on autograder feedback to direct their work. Ultimately, they are learning the grader, not learning the concepts" [3, p. 1062]. To mitigate this, the authors explored a penalty scheme where marks were deducted if a new submission resulted in a lower grade than a previous submission. The intent was to motivate students to test their own solutions rather than relying on the autograder feedback alone [3]. The authors reported that while the penalty scheme did cause some anxiety for students, it resulted in them testing their own solutions rather than always relying on the autograder.

2.2 Effects of feedback on behaviour

Procrastination [10, 21] and poor time management [39] have been identified as key problems for novice programmers and are indicative of inadequate self-regulatory skills [33]. There has been some research into the efficacy of interventions designed to reduce procrastination. Regular automated emails reminding students of impending deadlines were shown to have some limited success [15]. One study found that time management related visualisations mostly helped lower performing students, and that specific visualisations could even have adverse effects [24].

Most educators believe that encouraging students to start work early improves performance and research supports this [2, 18, 20, 42]. However, other research has shown that the time a student starts has very little effect on assessment outcomes [31]. Overall, there is more evidence supporting the premise that starting an assessment early has a positive influence on performance.

A number of interventions have been reported that provide assessment structures that encourage students to start early and work consistently. The use of mandatory staged problem solving sessions was found to result in engaged students who started their programming assignments earlier [42]. Irwin and Edwards used an energy bar, similar to those found in mobile games, in an effort to get students to spread their work out over the entire period of a programming assessment [26]. Students could submit their work for automated feedback as many times as they wished but each submission resulted in a loss of energy which would slowly regenerate over time. The goal of this gaming mechanism was to encourage students to begin their work earlier, however in practice students simply made fewer submissions. A similar mechanism was used by Indriasari et al. to influence students to complete peer code reviews ahead of a deadline [25]. Spacco et al. allowed students to submit their program code for testing, but in doing so they would consume a 'token' which would regenerate after 24 hours [37]. Both the energy bar and token systems were designed to limit the frequency with which students could submit to the autograder, yet students retained the freedom to submit at any time. Bouvier et al. describe a different approach, involving 'overnight feedback', for encouraging students to start early [5]. Students received feedback on a fixed schedule, once per night, over the course of a programming assignment. The authors report that students found the feedback helpful, and observed that fewer late submissions were made after the final deadline.

A recent paper on text comprehension examined two different assessment feedback scheduling approaches [7]. The assessment involved reading some text and then answering 12 questions about the text. Students were randomly allocated into one of three groups: no feedback (a control group), immediate feedback, or delayed feedback. Students in the immediate feedback group received feedback on a by-question basis whereas the delayed feedback group received feedback only after a block of four questions had been completed. They found that both groups of students who received feedback

read the text faster than the control group because the students knew they would have the opportunity to correct their answers based on the feedback and resubmit. The authors also reported that while feedback enhanced answer accuracy, there was no advantage in delayed feedback over immediate feedback. While this study is not situated in the context of novice programming, its findings are interesting and relevant to the mechanisms explored here.

In this paper, we describe our experiences using both a *scheduled feedback* system (where students receive feedback only if they submit their work prior to fixed scheduled deadlines) and an *immediate feedback* system (where students receive feedback instantly when submitting their work, but accrue penalties for over-reliance on the autograder). While both types of feedback have been utilised in prior studies, our work is novel in comparing the two types in two subsequent iterations of the same course, which can help in understanding their effects on students' submission behaviour. We compare the two approaches to see when students choose to make their submissions under each feedback system, and we explore these effects in detail for 'at-risk' students who have failed an earlier module in the course. We also report differences in student perceptions of the two approaches by looking at data from end-of-course satisfaction surveys.

3 METHODS

3.1 Course context

We explore the submission behaviour of students working on an end-of-course programming project in the final weeks of a 12-week introductory course. This course is a compulsory course taken by students in the first-year of an engineering program at a mid-sized urban university in Australasia. We compare two offerings of the same course in consecutive years (2019 and 2020), in both cases taught by the same instructor. We first briefly describe the nature of the programming projects, and then compare the two different feedback approaches – scheduled and immediate – used for the projects in each year.

3.2 Projects: text-based, grid-based

The end-of-course project contributes 12% towards a students' final grade and uses the C programming language. In both 2019 and 2020, students had 20 days to complete the project from the time that it was published. In both years the project was an ASCII text-based console program, which simulated a unique variation of some common strategy puzzle game involving a 2-dimensional grid. In 2019, the project involved implementing a version of the game Sokoban[1], in which a character moves around a grid and pushes objects to position them. In 2020, the game was a version of the sliding block puzzle Rush Hour[2], in which vehicles are moved to make way for a goal vehicle to leave the grid. Figure 1 shows screenshots of the ASCII-based interface for these two projects.

The projects in both years were of a similar level of difficulty, and both offerings were taught by the same instructor. Key dates for the two projects are shown in Table 1. In both years, students could receive feedback on their projects before the final deadline and make resubmissions. In 2019, two interim deadlines were scheduled, and

[1]https://en.wikipedia.org/wiki/Sokoban
[2]https://en.wikipedia.org/wiki/Rush_Hour_(puzzle)

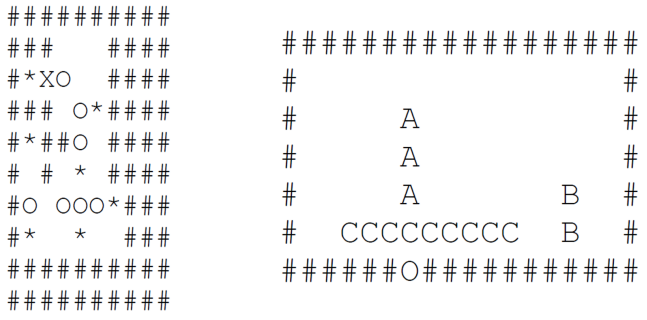


Figure 1: Examples showing the two ASCII-based projects. A game inspired by Sokoban (left) and Rush Hour (right)

Table 1: Key dates and deadlines for each project

Feedback	Event	Date & Time	Day
Scheduled (2019)	Project published	7 Oct	0
	Interim deadline #1	19 Oct 1:00pm	12
	Interim deadline #2	24 Oct 7:00am	17
	Final deadline	27 Oct 11:59pm	20
Immediate (2020)	Project published	11 Oct	0
	Grading period begins	27 Oct 1:00pm	16
	Final deadline	31 Oct 11:59pm	20

students could receive feedback on their work at the time of those deadlines. In 2020, students could receive immediate feedback on their work during a 'grading period' which lasted for about 5 days. Each project was divided into smaller tasks which could be graded independently, and if all tasks were completed successfully then the program would behave as specified. We now describe the two different feedback approaches used for each project.

3.3 Feedback schemes

3.3.1 Scheduled feedback. In 2019, two interim deadlines were set prior to the final project deadline. Students could submit their project – in any state of completion – to an online grading tool. Submitting work prior to these interim deadlines was optional (ungraded), but students who did would receive automated feedback on the work they submitted as soon as the interim deadline was reached. We refer to this approach as *scheduled* feedback, given that the feedback was generated only at the scheduled time of the interim deadlines.

When automated feedback was generated, each task (out of the 10 tasks for the project) was graded independently, and students received feedback by email that indicated what proportion of the test cases they passed for each task. Syntax errors and runtime errors were also highlighted on the report when applicable. The same test cases were used for these interim deadlines as for the final deadline.

3.3.2 Immediate feedback. In 2020, an automated grading tool was used to provide feedback to students, however it provided feedback immediately at the time code was submitted. Students could submit any task, one at a time, and receive feedback on the task in the form

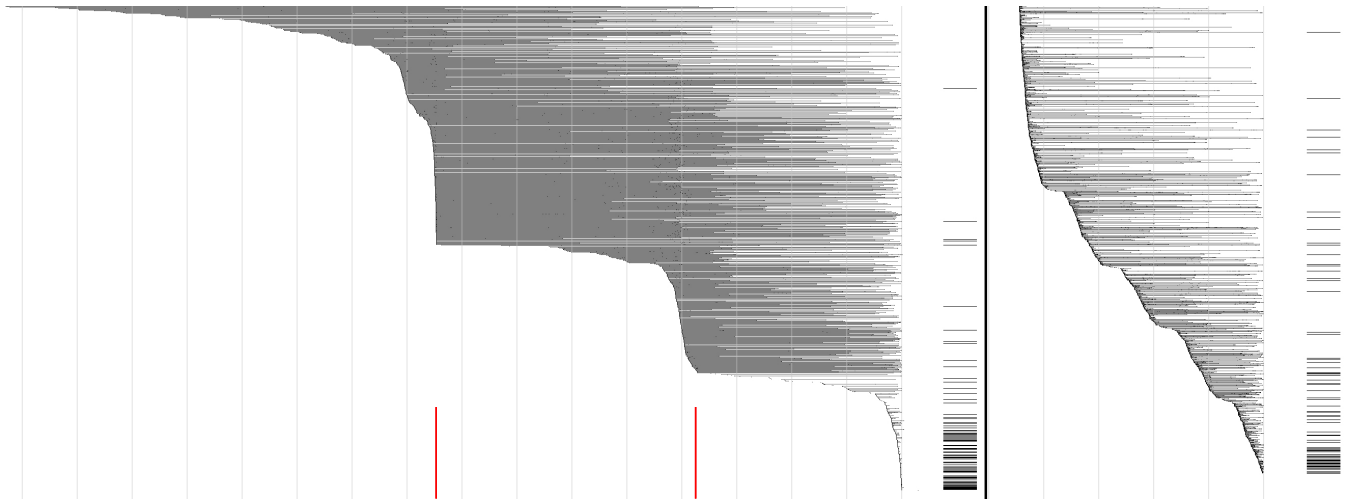


Figure 2: Submission patterns of students in the 2019 course (left; scheduled feedback) and 2020 course (right; immediate feedback). The two red bars denote the interim deadlines in 2019. The x-axis represents time, with vertical lines denoting midnight (the rightmost vertical line on each plot is the final deadline). Each row illustrates the submission behaviour of one student (a horizontal line connects their earliest and latest submission), sorted chronologically based on their earliest submission. Hatch marks on the right side of each plot denote students that ultimately failed the project.

of the proportion of test cases that passed, as before. To discourage students from over-reliance on this *immediate* feedback tool, two strategies were used. Firstly, it was only available for a short period, in the five days prior to the final project deadline. Secondly, a penalty scheme was introduced whereby the first two submissions for any task incurred no penalty, but every subsequent submission incurred an additional 10% penalty (up to a maximum of 70%).

3.4 Data analysis

Our analysis includes data for all students who made at least one project submission. In 2019, a total of 952 students submitted a project whereas in 2020, 918 students submitted a project. All of the submissions that students made were timestamped and we use these times to generate a visualisation of the submission behaviours across the two years. In addition, we use the final score that each student earned on the project to understand how submission times correlate with project performance.

We are also interested in the cohort of students who are deemed 'at-risk' of failing the course. The 12-week course is taught in two modules (6 weeks each), and we use student performance on the first module to identify at-risk students. All students who failed (i.e. scored <50%) on the first module are classified as at-risk. A total of 85 students in the 2019 course, and 65 students in the 2020 course, are in this category.

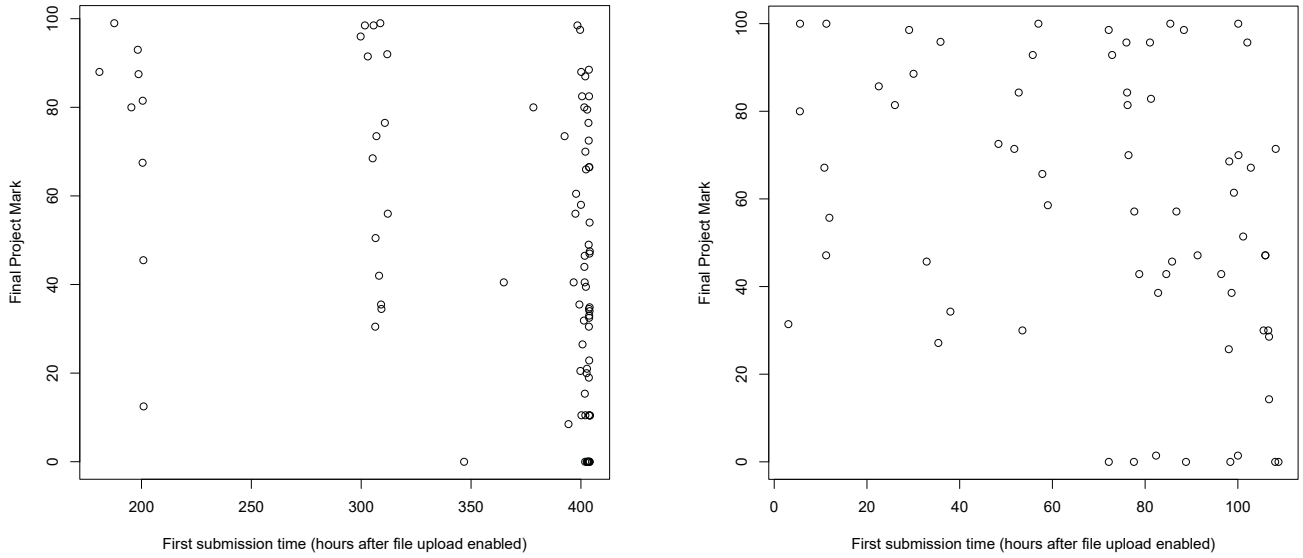
Finally, we examine data from end-of-course surveys to explore how students perceived the scheduled and immediate feedback. In both courses, students were invited to complete an end-of-course teaching satisfaction survey in line with standard institutional policy. On each survey, students read and responded to 10 statements regarding course satisfaction using a 5-point Likert-scale, and could provide open-ended comments to describe aspects of the course they found helpful or require improvement. We examine quantitative scores for the question "I received helpful feedback on my learning progress", and we look at responses to the open-ended questions for any statements related to project feedback.

4 EXPERIENCES WITH SCHEDULED AND IMMEDIATE FEEDBACK

4.1 Students' submission behaviour

Figure 2 shows students' submission patterns for the 2019 (scheduled feedback) and the 2020 (immediate feedback) courses. The left side of the figure shows the patterns for the 2019 course with the two optional interim feedback deadlines (shown in red) and the right side shows the patterns for the 2020 course where immediate feedback with penalties was available. Time increases going to the right, and each vertical line denotes midnight. Each row of the figures represents the submission behaviour of a single student in the course. All submissions are shown as individual dots, and to ease visibility, the left most (i.e. earliest) and rightmost (i.e. latest) submissions for each student are connected by a horizontal line. In both figures, students have been sorted chronologically, based on the time of their earliest submission. The short horizontal hatch marks next to the lines on the very right of each figure denote the students who failed the project, i.e. scored less than 50%.

From Figure 2, it is evident that the optional interim feedback deadlines affected students' submission behaviour. For the 2019 course, where these optional feedback deadlines were available, there's a clear stair-like pattern that shows many students submitting their work very close to the deadlines, and very few submitting their work immediately after such a deadline. In contrast, for the



(a) 2019 course **(b) 2020 course**

Figure 3: At-risk students' earliest submissions plotted against their final project marks. The x-axis shows the time (with respect to when the autograder was enabled) when the student first submitted their project for feedback. The y-axis shows final project marks. In 2019 (left; scheduled feedback), there were two optional interim feedback deadlines. In 2020 (right; immediate feedback) students received feedback immediately.

2020 course with immediate feedback, we see a less prominent stair-like pattern, indicating that students tended to make submissions that were spread more evenly over the course of the day.

Looking at the hatch marks that represent students who failed the project, we see a very clear pattern that students who initially submitted their work closer to the final deadline were more likely to fail the project in both courses. Interestingly, in the 2019 course, we see that almost no students who utilized the first of the interim deadlines ended up failing the project. This pattern is less prominent in 2020, where several students who submitted during the first 24 hours of the grading period still ended up failing the project.

Another pattern visible in the figures is that activity for the 2020 course seems more sparse. This is due to the fact that less time tends to elapse for students in 2020 between the very first and the very last submission they make. This is expected, given that students in 2020 received feedback immediately and could therefore attend to any issues straight away. They could work on the project until it was solved correctly. In comparison, issues highlighted by the first interim deadline for students in 2019 did not need to be addressed until the next interim deadline.

4.2 Feedback approach and at-risk students

We are particularly concerned about how the feedback approaches might affect at-risk students. Thus, we examined submission patterns and final project marks for students deemed to be at-risk based on having failed the first module in the course. These students are at high risk of failing the course, given that course grades are based on a combination of the first and second module results. Figure 3 plots the time at which at-risk students first submitted their programs for feedback and the final marks they ultimately received for the project.

From Figure 3 we see that, similar to the data for the whole student population shown in Figure 2, the submission behaviour of the at-risk students was clearly influenced by the interim deadlines. We see three vertical bands on the plot, with the rightmost band being the most dense. This illustrates that although some at-risk students submitted prior to the first and the second interim deadlines, most students in this category made their very first submission (and thus received no feedback they could use to improve their work) just prior to the final deadline.

Of primary interest is the relationship between starting (and submitting) work early and subsequent performance. This relationship is clearer in the 2019 course – the mean project mark for students who submitted prior to the first and second interim deadline (the left and middle bands in Figure 3a) is 72.7% and 69.5%. In comparison, the mean score for students who only submitted prior to the final deadline is 40.2%. In the 2020 data, the relationship is less clear. However, one interesting observation is that there is a sizeable group of students near the bottom right of Figure 3b who received a score of 0 (or near 0) for the project.

One interpretation of these findings is that the scheduled feedback approach is more helpful for at-risk students. As with other immediate feedback schemes (e.g. [26]), the desired change in behaviour is implicit. Students take on the responsibility of managing their own time, and must determine for themselves when to start their work so that enough time is left to respond to the feedback

they receive. On the other hand, the scheduled interim deadlines send a much more explicit signal around timing – students must start early enough to meet the scheduled deadlines if they want to receive feedback. Given the evidence that at-risk students perform much better when they do receive such feedback, approaches that more explicitly encourage them to start earlier may be advisable.

4.3 Student perceptions of feedback

In the preceding sections we focused on submission patterns and their relationship to project outcomes. In both courses, students completed an end-of-course satisfaction survey, and examining the responses to this survey reveals an interesting difference in the way students perceived the scheduled and immediate feedback.

On each survey, students read and responded to statements regarding course satisfaction. These statements included one related to 'feedback' specifically: "I received helpful feedback on my learning progress". Traditionally, this is one of the lowest scoring statements for courses in the studied context. Indeed, it was the second lowest scoring statement on both surveys despite the fact that the automated approaches we explored do provide feedback to students that would not ordinarily be available in the course. The final question was an 'overall' assessment of the course: "Overall, I was satisfied with the quality of this course".

There was a marked difference in student responses to the 'feedback' question across the two courses. For the 2019 course (scheduled feedback), 287 students responded to the survey and the percentage of responses that were in agreement, neutral or in disagreement with the 'feedback' statement was 87.5%, 8.4% and 4.2%. In comparison, for the 2020 course (immediate feedback), 250 students responded and the percentage of responses that were in agreement, neutral or in disagreement with the 'feedback' statement was 71.7%, 19.6% and 8.7%, indicating much lower levels of satisfaction. Some of this may be due to negative perceptions of the penalty scheme used in 2020 as prior work has found that students perceive submission-limiting approaches negatively [26]. However, given that they could request feedback on a task-by-task basis, and received two non-penalised submissions for each task, the amount of feedback available without penalty was greater than for students in 2019. It is also possible that some of this effect is due to fewer in-person interactions in 2020 due to the COVID-19 pandemic, however the 'overall' course satisfaction scores are similar in both years (92.3% agreement vs 89.2% agreement in 2019 and 2020 respectively). Therefore, it may be that the scheduled feedback provided in 2019 had a more pronounced impact on students' perceptions of feedback provided in the course.

This is further supported by the open-ended feedback provided by students on the course satisfaction surveys in both years. In 2019, around 5% of students (14 out of 287) wrote a comment that explicitly mentioned the usefulness of the scheduled feedback for the project. In 2020, only 2 students (fewer than 1%) provided comments to this effect, and these two comments identified the project as only one assessed activity where immediate feedback was useful (i.e. *"The instantaneous feedback for the labs was really helpful. Likewise with the project"*). In contrast, the comments provided in 2019 were much more enthusiastic about the value of the scheduled feedback. Several comments indicated that the feedback helped students improve their final mark for the project:

- *"I really like the early feedback... It was really helpful in making sure all my code worked correctly."*
- *"i really liked how you gave us the opportunity to submit projects early, really helped me figure out where i went wrong and where to improve"*
- *"really liked the early submission thing since I thought my code was fine."*

4.4 Limitations

In addition to the different feedback approaches, there were other factors that may have influenced students' project outcomes and perceptions in 2019 and 2020. For example, although the two course offerings were similar in 2019 and 2020, using the same course material and taught by the same instructor, there were slight differences between the projects. In 2019, there were ten tasks in the project, and all ten tasks were graded at the same time (students would submit a single file prior to an interim deadline to receive feedback on all tasks they had attempted). In 2020, there were only seven tasks in the project, and students could grade these tasks separately (essentially submitting the relevant functions one at a time). However, the complexity of the project and the expectation of the time that students should spend on the project was similar across both years. This is empirically supported by the fact that the distribution of final project marks were almost identical across the two years. The mean (and SD) scores for the projects in 2019 and 2020 were 86.8% (21.6) and 88.7% (20.5), respectively.

Additionally, students in the 2019 course were not affected by the COVID-19 pandemic, whereas students in the 2020 course experienced some disruption to in-person teaching, as approximately half of the semester was conducted online. Nevertheless, the similarities in project scores shown above suggests this may not have been a significant factor.

5 CONCLUSIONS

In this work, we report our observations on student submission behaviours in two courses where different approaches to feedback were employed. In the first approach, students could submit work early to meet two optional interim deadlines. As soon as these scheduled deadlines passed, the authors of any submissions received automated feedback on their work. In the second approach, students could submit work at any time and receive immediate feedback, but with increasing penalties to discourage excessive submissions.

We found that both the whole student population, and students at high risk of failing, exhibited markedly different submission patterns for the two feedback approaches. When immediate feedback was available, submissions were more evenly spread out over time, whereas students worked very closely to the published deadlines when feedback was scheduled. Interestingly, student perceptions of the scheduled feedback approach were more positive. In both cases, the later a student made their first submission, the more likely they were to fail the project. These results highlight the need to find better ways of encouraging students to start work early. In particular, future work should seek to identify interventions that are effective at nudging a greater proportion of at-risk students towards starting work earlier.

ACKNOWLEDGMENTS

We are grateful for the grant by the Media Industry Research Foundation of Finland which partially funded this work.

REFERENCES

[1] Kirsti M Ala-Mutka. 2005. A survey of automated assessment approaches for programming assignments. *Computer science education* 15, 2 (2005), 83–102.

[2] Tapio Auvinen. 2015. Harmful Study Habits in Online Learning Environments with Automatic Assessment. In *2015 International Conf. on Learning and Teaching in Computing and Engineering*. 50–57.

[3] Elisa Baniassad, Lucas Zamprogno, Braxton Hall, and Reid Holmes. 2021. STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 1062–1068.

[4] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In *Proceedings of the Working Group reports on Innovation and Technology in Computer Science Education*. 177–210.

[5] Dennis Bouvier, Ellie Lovellette, and John Matta. 2021. Overnight Feedback Reduces Late Submissions on Programming Projects in CS1. In *Australasian Computing Education Conf.* 176–180.

[6] Brent J. Bowman and William A. Newman. 1990. Software Metrics as a Programming Training Tool. *J. Syst. Softw.* 13, 2 (Oct. 1990), 139–147.

[7] Carmen Candel, Eduardo Vidal-Abarca, Raquel Cerdán, Marie Lippmann, and Susanne Narciss. 2020. Effects of timing of formative feedback in computer-assisted learning environments. *J. of Computer Assisted Learning* 36, 5 (2020), 718–728.

[8] Rachel Cardell-Oliver. 2011. How can software metrics help novice programmers?. In *Proceedings of the Thirteenth Australasian Computing Education Conf.-Volume 114*. 55–62.

[9] Rachel Cardell-Oliver, Lu Zhang, Rieky Barady, You Hai Lim, Asad Naveed, and Terry Woodings. 2010. Automated feedback for quality assurance in software engineering education. In *2010 21st Australian Software Engineering Conf.* IEEE, 157–164.

[10] Sophie H. Cormack, Laurence A. Eagle, and Mark S. Davies. 2020. A large-scale test of the relationship between procrastination and performance using learning analytics. *Assessment & Evaluation in Higher Education* 45, 7 (2020), 1046–1059.

[11] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. Codewrite: supporting student-driven practice of java. In *Proceedings of the 42nd ACM technical symposium on Computer Science Education*. 471–476.

[12] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. Understanding the Syntax Barrier for Novices. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 208–212.

[13] Paul Denny, James Prather, and Brett A Becker. 2020. Error message readability and novice debugging performance. In *Proceedings of the 2020 ACM Conf. on Innovation and Technology in Computer Science Education*. 480–486.

[14] Paul Denny, Jacqueline Whalley, and Juho Leinonen. 2021. Promoting Early Engagement with Programming Assignments Using Scheduled Automated Feedback. In *Australasian Computing Education Conf.* 88–95.

[15] Stephen H Edwards, Joshua Martin, and Clfford A Shaffer. 2015. Examining classroom interventions to reduce procrastination. In *Proceedings of the 2015 ACM Conf. on Innovation and Technology in Computer Science Education*. 254–259.

[16] Stephen H Edwards and Krishnan Panamalai Murali. 2017. CodeWorkout: short programming exercises with built-in data collection. In *Proceedings of the 2017 ACM Conf. on Innovation and Technology in Computer Science Education*. 188–193.

[17] Stephen H Edwards and Manuel A Perez-Quinones. 2008. Web-CAT: automatically grading programming assignments. In *Proceedings of the 13th annual conf. on Innovation and technology in computer science education*. 328–328.

[18] Stephen H Edwards, Jason Snyder, Manuel A Pérez-Quiñones, Anthony Allevato, Dongkwan Kim, and Betsy Tretola. 2009. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop*. 3–14.

[19] Tommy Färnqvist and Fredrik Heintz. 2016. Competition and feedback through automated assessment in a data structures and algorithms course. In *Proceedings of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*. 130–135.

[20] James B Fenwick Jr, Cindy Norris, Frank E Barry, Josh Rountree, Cole J Spicer, and Scott D Cheek. 2009. Another look at the behaviors of novice programmers. *ACM SIGCSE Bulletin* 41, 1 (2009), 296–300.

[21] Yoshiko Goda, Masanori Yamada, Hiroshi Kato, Takeshi Matsuda, Yutaka Saito, and Hiroyuki Miyagawa. 2015. Procrastination and other learning behavioral types in e-learning and their relationship with learning outcomes. *Learning and Individual Differences* 37 (2015), 72–80.

[22] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. 2010. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli calling international conf. on computing education research*. 86–93.

[23] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on Working Group Reports* (2015), 41–63.

[24] Kalle Ilves, Juho Leinonen, and Arto Hellas. 2018. Supporting self-regulated learning with visualizations in online learning environments. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 257–262.

[25] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2021. Improving Student Peer Code Review Using Gamification. In *Australasian Computing Education Conference* (Virtual, SA, Australia) *(ACE '21)*. Association for Computing Machinery, New York, NY, USA, 80–87.

[26] Michael S Irwin and Stephen H Edwards. 2019. Can Mobile Gaming Psychology Be Used to Improve Time Management on Programming Assignments?. In *Proceedings of the ACM Conf. on Global Computing Education*. 208–214.

[27] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)* 19, 1 (2018), 1–43.

[28] Pardha Koyya, Lee Young, and Jeong Yang. 2013. Feedback for Programming Assignments Using Software-Metrics and Reference Code. *ISRN Software Engineering* (2013).

[29] Juho Leinonen, Francisco Enrique Vicente Castro, and Arto Hellas. 2021. Does the Early Bird Catch the Worm? Earliness of Students' Work and its Relationship with Course Outcomes. In *Proceedings of the 26th ACM Conf. on Innovation and Technology in Computer Science Education V. 1*. 373–379.

[30] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An evaluation of the impact of automated programming hints on performance and learning. In *Proceedings of the 2019 ACM Conf. on International Computing Education Research*. 61–70.

[31] Keir Mierle, Kevin Laven, Sam Roweis, and Greg Wilson. 2005. Mining student CVS repositories for performance indicators. *ACM SIGSOFT Software Engineering Notes* 30, 4 (2005), 1–5.

[32] Nick Parlante. 2007. Nifty Reflections. *SIGCSE Bull.* 39, 2 (June 2007), 25–26.

[33] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. 2020. What Do We Think We Think We Are Doing?: Metacognition and Self-Regulation in Programming. In *Proceedings of the 2020 ACM Conf. on International Computing Education Research*. ACM, New York, NY, USA, 12 pages.

[34] A. Ramaprasad. 1983. On the definition of feedback. *Behavioral Science* 28, 1 (1983), 4–13.

[35] Paul Ramsden. 1992. *Learning to Teach in Higher Education*. Routledge, London.

[36] Jaime Spacco, Paul Denny, Brad Richards, David Babcock, David Hovemeyer, James Moscola, and Robert Duvall. 2015. Analyzing student work patterns using programming exercise data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 18–23.

[37] Jaime Spacco, Davide Fossati, John Stamper, and Kelly Rivers. 2013. Towards improving programming habits to create better computer science course outcomes. In *Proceedings of the 18th ACM Conf. on Innovation and technology in computer science education*. 243–248.

[38] Claudia Szabo, Nickolas Falkner, Antti Knutas, and Mohsen Dorodchi. 2018. Understanding the effects of lecturer intervention on computer science student behaviour. In *proceedings of the 2017 ITiCSE conf. on working group reports*. 105–124.

[39] Rebecca Vivian, Katrina Falkner, and Nickolas Falkner. 2013. Computer science students' causal attributions for successful and unsuccessful outcomes in programming assignments. In *Proceedings of the 13th Koli Calling International Conf. on Computing Education Research*. 125–134.

[40] Jacqueline L Whalley and Anne Philpott. 2011. A unit testing approach to building novice programmers' skills and confidence. In *Proceedings of the Thirteenth Australasian Computing Education Conf.*, Vol. 114. 113–118.

[41] Dylan Wiliam. 2011. What is assessment for learning? *Studies in educational evaluation* 37, 1 (2011), 3–14.

[42] Salla Willman, Rolf Lindén, Erkki Kaila, Teemu Rajala, Mikko-Jussi Laakso, and Tapio Salakoski. 2015. On study habits on an introductory course on programming. *Computer Science Education* 25, 3 (2015), 276–291.