Does the Early Bird Catch the Worm? Earliness of Students' Work and its Relationship with Course Outcomes

Juho Leinonen
juho.leinonen@helsinki.fi
University of Helsinki
Helsinki, Finland

Francisco Enrique Vicente Castro
fcastro@cs.umass.edu
University of Massachusetts Amherst
Amherst, Massachusetts, USA

Arto Hellas
arto.hellas@aalto.fi
Aalto University
Espoo, Finland

ABSTRACT

Intuitively, it seems plausible that students who start their work earlier and work on more days than their peers should perform better in any course. But does the early bird really catch the worm? In this article, we examine introductory programming students' time management behavior as evidenced by data collected from a programming environment. We analyze: 1) the earliness of students' work, i.e. when they start working on their course assignments, 2) the number of days students work on course assignments, and 3) the relationship between earliness, the number of days worked, and course outcomes. Our results provide further support for the notion that, on average, students who start working on course assignments early perform slightly better in the course. At the same time, we found that starting early does not necessarily mean that students work on more days, and that starting early and working on many days does not necessarily mean that students get better grades. In addition, some students who start working early on the assignments in the first weeks of the course seem to start delaying when they begin working on assignments as the course progresses, while other students seem to be able to continue starting early throughout the course.

CCS CONCEPTS

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

KEYWORDS

time management, earliness, procrastination, self-regulation, keystroke analysis, cs1, introductory programming, novice programmer, learning analytics, educational data mining

ACM Reference Format:
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 *26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2021), June 26–July 1, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456383

ITiCSE 2021, June 26–July 1, 2021, Virtual Event, Germany

ACM ISBN 978-1-4503-8214-4/21/06…$15.00
https://doi.org/10.1145/3430665.3456383

1 INTRODUCTION

Research into earliness of students' work (or the lack of it) in computing education research has provided some insight into when students work and how the scheduling of their work relates to course outcomes. For example, Edwards et al. [10] found that students who start their project work early often achieved higher project scores than those who did not, and Auvinen et al. [1] observed that individual time management practices may also influence time management practices in group work. As such, it is not surprising that researchers have sought ways to improve students' time management practices (e.g. [15]).

Although the generic trend seems to be that starting to work on course assignments early leads to better course outcomes, and that starting early is likely linked with working on more days [7], the underlying factors that contribute to these effects are not clear. At times, results from studies that have sought to determine these underlying factors may have even been contradictory. As an example, when considering students' metacognitive strategies, measured using the widely-used MSLQ questionnaire [26], Bergin et al. [2] and Watson et al. [32] had noticeable differences in their research outcomes, possibly highlighting the effect of contextual differences.

In this work, we study students' time management as evidenced by log data gathered from their programming process in an introductory programming course, exploring how observed time management behaviors relate to course outcomes. In particular, we are interested in the *earliness* of students' work—i.e. when they start to work on course assignments—and the *spacing* of students' work—i.e. how many days they work on course assignments—and the relationship of these two variables when considering course outcomes. We also consider the possible effect of the context on the study outcomes, and point out the need for further insight into why students behave as they do.

The closest matches to our work are studies where time management of students, and especially earliness of their work, has been studied based on log data such as the studies by Edwards et al. [10], Watson et al. [32], and Spacco et al. [30]. We build upon that work by including students' *spacing* of their work, i.e. on how many days students work on assignments, into the analysis.

This article is organized as follows: in the following Section 2, we go over work related to time management, focusing on prior studies within computing education. In Section 3, we describe the pedagogical context of the study, our data and research methods, and our research questions. We describe the results of our analysis in Section 4, which we discuss further in Section 5. Lastly, we summarize our overall study and findings, and outline possible future work in Section 6.

2 RELATED WORK

Within computing education research, earliness of students' work and students' time management practices have been broadly studied using two approaches or a combination of them: (1) using scales and surveys that quantify time management (e.g. [2, 20, 21, 32]), and (2) using data collected from learning environments (e.g. [10, 17–19, 30]).

When using surveys to quantify metacognitive strategies (including planning and scheduling of work measured, for example, with the MSLQ questionnaire [26]), researchers have found contradictory evidence on the effect of metacognitive strategies on course outcomes. For example, Bergin et al. [2] observed that metacognitive strategies had a medium effect size on course outcomes, while Watson et al. [32] and Longi [21] did not identify statistically significant correlations. Leppänen et al. [20] suggested that such discrepancies could be related to students answering based on what they think they are doing instead of what they are actually doing. Additional evidence on this would be needed, however—in general, there are rather few studies on metacognition and self-regulation within computing education research [27].

The lack of time management skills has been linked to lower productivity and lower academic performance [22], and interest in using students' self-regulation for predicting performance is on the rise [13]. Poor time management skills are also linked with counterproductive study strategies such as plagiarism [5], as well as stress and anxiety [24]. At an extreme, poor time management may also include procrastination, i.e. delaying tasks that need to be completed to the extent that they can no longer be completed at an expected level, or at all [12]. Not all delaying of tasks is procrastination, however, as one may delay the starting of a task to think about how it should be approached, such as in the case of incubation [28].

Good time management among students may manifest through, for example, students starting to work on their assignments early. Using data collected from learning environments, Edwards et al. [10] observed that students who start working on their projects early tend to achieve better scores on the projects than those who start working on the projects late. Similar observations, also related to correctness of submitted projects, were reported by Martin et al. [23] and Denny et al. [8]. These results may be linked with students starting early potentially spending more time on the projects, as suggested by Denny et al. [7], although there are also studies that suggest that starting early also leads to submitting work early [16, 23].

For learning, however, it is not always ideal to submit the work early, even if the work would be started early. The *spacing effect* [9, 14] stipulates that learning is enhanced when study sessions are spaced out. That is, one should prefer dividing work over multiple study sessions (e.g. days) instead of "cramming" all the work in the same space. Much of the work on the spacing effect has been done with recall and there are issues that complicate the interpretation of spacing studies [6]. It is also not clear whether the effect should hold in programming. Indeed, there are some studies that suggest that the spacing effect does not hold for complex motor skills such as learning to play the piano [33].

Our current project builds on this prior work in several ways. First, we examined CS1 students' time management through an analysis of fine-grained log data collected from an IDE used in a CS1 course. We looked at when students started work on their programming assignments (i.e. *earliness*) and their active periods of work prior to the assignment deadlines (i.e. *spacing* of their work). These allow us to get a sense of how students were using their time prior to the deadline to work on course assignments. We also explored potential relationships between students' earliness and work-spacing, and their overall performance (grade) on the course.

3 METHODOLOGY

3.1 Course Context

This study was conducted in an introductory programming course (CS1) offered by the University of Helsinki. The university follows a quarterly system: a quarter is 8 weeks, with 7 weeks for teaching and a final week for exams. The introductory programming course uses Java as the programming language, teaches students the principles of composing solutions to programming problems, and starts with a procedural programming approach that introduces students to standard input and output, variables, conditional statements, loops, functions, lists, and maps. Students then proceed with the basics of object-oriented programming, with a focus on representing data as objects and using objects for separating concerns (e.g. UI, logic, data).

3.1.1 ***Pedagogy.*** The course uses an online textbook with embedded videos, questionnaires, and programming assignments, and follows a *many small assignments* approach, where students work on multiple assignments whenever they learn a new topic. When learning a new topic, students often first see a few multiple-choice questions and a few programming assignments, which may combine into larger programs, seeking to implicitly teach program (de)composition. The course has a total of 147 programming assignments; while the majority of the assignments are small (e.g. printing if an input value is negative or positive, solving the rainfall problem [29], etc.), the latter weeks of the course also contain relatively complex and open-ended assignments (e.g. Tournament Uno by Stephen Davies [25]).

3.1.2 ***Coursework, Exams, and Grading.*** Coursework (questionnaires and programming assignments) are automatically assessed, and the course has weekly deadlines. In the course under study, the deadline was at midnight on Mondays; the sole lecture of the week was on Tuesdays. Assignments were released a week before the deadline, except for the last week, where students had an additional week to work on the assignments. Due to technical issues with the automatic assessment system, students were given two extra days on week four to submit assignments. In addition to the feedback from the automatic assessment system, students can also attend feedback labs with TAs and course personnel (the course was run prior to the COVID-19 pandemic).

The course has two exams: a midterm exam, given on the third week, and a final exam, given on the final week. Exams are similar in content to the programming assignments of the course and students need to receive at least half of the exam points in order to pass the course. The final course grade is computed as a combination of the

exams and the coursework, where 70% of the overall points come from the course questionnaires and programming assignments, and 30% of the points come from the exams (10% from the midterm and 20% from the final exam). The course grade is on a six-point scale from 0 to 5. The highest grade is 5, which is achieved by collecting at least 91% of the overall points, while 70% of the overall points leads to the lowest passing grade of 1. Additionally, students are required to get at least half of the points from the final exam in order to pass the course.

As an alternative to passing the course by coursework and exams, the university offers the option of taking separate "alternate" exams. When taking an alternate exam, the course grade will be fully determined by the alternate exam (i.e. coursework is not counted towards the grade). While this helps in cases when students have overlapping exams, some students opt for this to attempt to gain a better grade (e.g. due to not completing sufficient coursework for their desired grade). Official transcripts do not contain details on failed courses.

Students enroll to the course at the end of the previous quarter, and joining in late is possible, although late submissions of assignments are not allowed. Students at University of Helsinki are allowed to "sample" the offered courses, which often results in more students attending the first week of the course than the subsequent weeks.

3.2 Data and Preprocessing

Students in the course use Test My Code [31], which is a customized IDE that supports downloading the programming assignments and submitting them for assessment. The IDE collects data about students' programming processes as they are completing assignments, such as students' timestamped submissions and keystrokes within the IDE. We focused our analyses on the timestamped data collected from the IDE and the students' final course grade. Overall, we analyzed data from 345 students in the course. Table 1 shows the number of programming assignments and the number of active students, i.e. students who had at least a single keystroke, for each week of the course. Only data from students who had provided consent to use their data for research was used.

Table 1: The number of programming assignments (*PAssigns*) and active students (*Students*) per week of the course.

Week	1	2	3	4	5	6	7	Overall
# PAssigns	32	24	30	25	10	16	10	147
# Students	318	312	302	298	271	272	255	345

3.3 Research Questions

In this work, we are interested in understanding how students' earliness of work and time management relates to course outcomes. We study this through the following research questions (*RQ*s):

RQ1 When do students start working on programming assignments and to what extent does their start time influence the number of days they take to work on their assignments?

RQ2 How does when students start working on programming assignments and the number of days students work on programming assignments relate to their performance and effort in the course?

For *RQ1*, we look at when students start working on weekly course assignments as evidenced by the first keystroke on the course assignments. Having a keystroke for an exercise indicates that students have downloaded the assignment and very likely read the assignment description. We also look at how many days the students work on the course assignments as evidenced by the number of distinct days when students submit at least a single assignment.

For *RQ2*, we study students' earliness on each course week and how many days the students work on the course assignments, and contrast these with the course outcomes as evidenced by the grade students got from the course.

4 RESULTS

We analyzed data (Section 3.2) from all 7 weeks of the course. While all our discussions of our analyses and findings cover all our data, we selected to present visualizations (Figures 1 to 4) from weeks 1 and 5 for the sake of space and because these best highlight our salient findings[1]. Other weeks of the course show similar trends.

4.1 Earliness and Number of Days Worked on Assignments

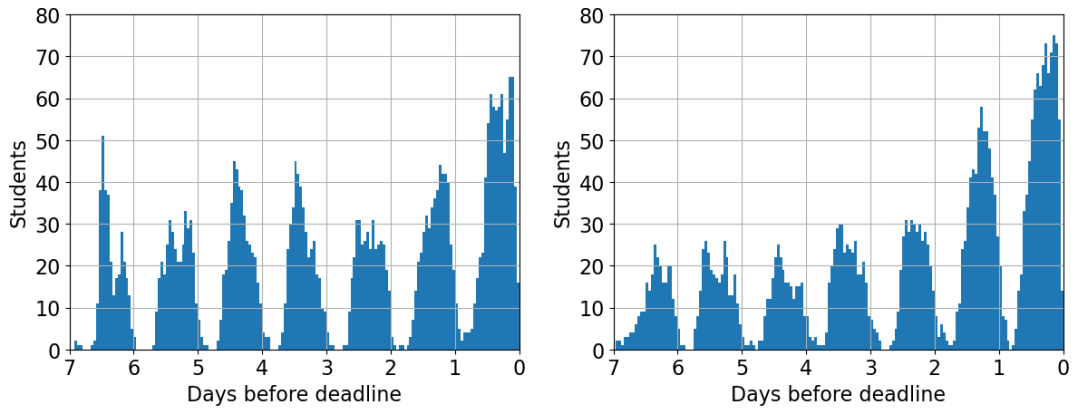
Figure 1 shows the number of active students at every hour of weeks 1 and 5 of the course. Looking at the figure, we see that in the first week of the course, the number of students working on each day of the course seems pretty equal, peaking at around 40 students in the most active hour in all of the days except the final day before the deadline, when the most active hours have slightly over 60 students active. Looking at the fifth week of the course, we see that fewer students are active in the first 5 days (peak activity has around 20-30 students active), and that the two last days have higher activity peaks (at 60 and almost 80 students active during peak hours for the second to last and the last day before the deadline respectively).

Figure 2 shows the relationship between when students started working on course assignments and how many days students worked on course assignments for weeks 1 and 5. We see that, interestingly, there is not a big difference between students who start very early (6 or 5 days before the deadline) and somewhat early (3 to 4 days before the deadline), and in both cases, the median number of work days is three. Those who start close to the deadline (1 day before or on the day of the deadline) have fewer days, but this is to be expected as those who, for example, start on the day of the deadline, only have a single work day left to work on their assignment.

4.2 Earliness, Number of Days Worked, and their Relation to Course Performance

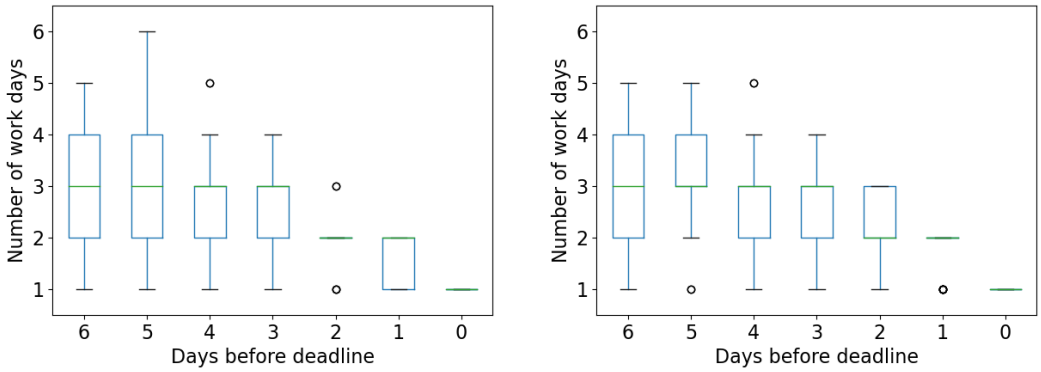
Figure 3 shows how students who started work a certain number of days before the deadline perform in the course. Here, students are split into three groups based on their final course grade. We categorize as *low* performers those who get either a 0 (fail) or a

[1]The visualizations for all weeks are at: https://github.com/mession/iticse-2021



(a) Week 1 **(b) Week 5**

Figure 1: Number of active students (per hour) on the days before the programming assignment deadline



(a) Week 1 **(b) Week 5**

Figure 2: When students started (*Days before deadline*) and how long they worked on their assignment (*Number of work days*)

1; as *medium* performers those who get a 2, 3, or 4; and as *high* performers those who get a 5. We see that for both of the weeks visualized, there is a clear trend (Pearson's $r = 0.23, p = 0.0003$ for week 1 and $r = 0.30, p < 0.0001$ for week 5) that those who started earlier performed better in the course. However, even of those who start one day before the deadline, most will get a 5, i.e. the best grade available. Only for those who start on the very last day, the day of the deadline, the most likely grade is 0 or 1. The trends are similar also for the weeks not visualized here.

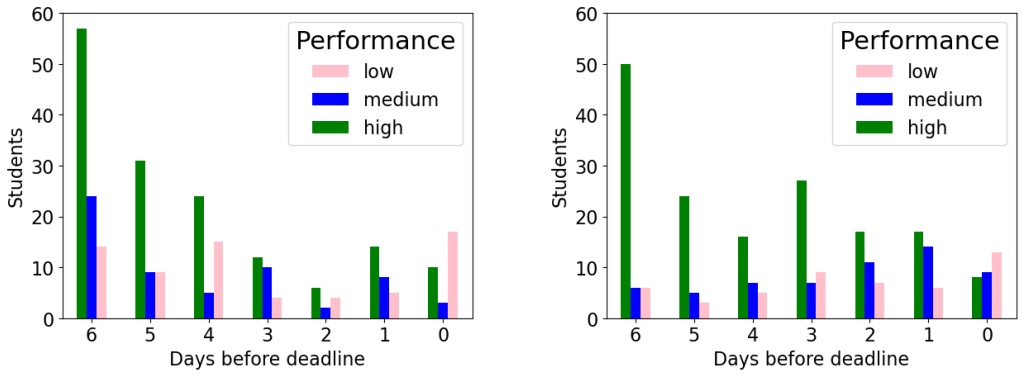
Figure 4 visualizes the average grade of students based on when they started working on their programming assignments and how many days they worked on their assignments. The average was calculated only for combinations where data from at least five students was available, and only for those who attended the final exam. We again see a clear trend that students who start work earlier perform better. Interestingly, for some cases, we note that when taking the number of work days into account, for students who start on the same day, those that work on fewer days seem to perform better compared to those who work on more days (e.g. on weeks 1 and 2).

5 DISCUSSION

5.1 Does Starting Early Mean Working More?

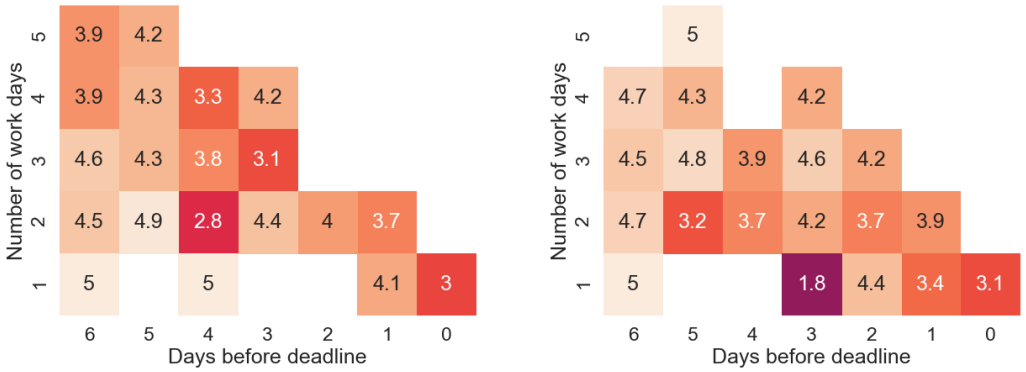
We found that most students start working on course assignments early. However, we still observe that the last day before the deadline is the most active (see Figure 1). Since our data is only quantitative in nature, we do not know why some students start late and why the last day is the most active. A possible explanation is that students procrastinate [12], but it is also possible that students start late because of the incubation effect [28], i.e. thinking about the problem before working on it.

We found that students who started earlier tend to work on more days (see Figure 2), which is not surprising – there is an implied relationship between the two as those who start earlier simply have a larger number of possible days when they can work on assignments. Interestingly, the differences between, for example, those who start on the first day that assignments are available, and those who start, for example, three days before the deadline, are not great. One possible explanation is that there is a "ceiling" as once students have completed all of the assignments, they will not



(a) Week 1 (b) Week 5

Figure 3: Number of students and when they started working on their assignments, binned by final course grade: low (0 and 1), medium (2, 3, and 4), high (5).



(a) Week 1 (b) Week 5

Figure 4: Heatmaps indicating the average course grade of students (values inside squares) by their start times (x-axis) and how long they worked on their assignment (y-axis). Darker shades indicate lower grades. (Lowest grade: 0, highest grade: 5).

accrue more work days. This has a few implications. Firstly, simply getting students to start earlier might not mean that they will work on more days. This suggests that interventions aimed at simply getting students to work earlier [15, 16, 23] might not be as useful as one might expect. On the other hand, this could also mean that those interventions could be effective even if deployed only, for example, three or four days before the deadline.

5.2 Early start = Better grade: Not that simple

In our context, there seems to be a trend that students who start earlier end up with better grades in the course (see Figures 3 and 4). However, while the proportion of students getting a high grade diminishes for those who start work closer to the deadline, even for those who start just one day before the deadline, the most common grade is 5, which is the highest available grade. This suggests that, for example, predicting performance purely based on when students start working on assignments might be ineffectual. One possible explanation for why we see this effect is that some students who perform well and start late could have had previous programming experience and thus know that they can start late and still complete all the assignments. Additionally, since the deadline was on Mondays, those who start one day before the deadline start work on a Sunday. It is likely that students have more time to work on assignments on Sunday compared to weekdays, which could partly explain why students starting just one day before the deadline are still able to achieve high grades.

Prior research suggests that students who start early perform better [10] and that because of the spacing effect [9, 14], students who split work over multiple days should perform better than those who work on fewer days. Interestingly, we found that when examining students grouped by how many days before the deadline they started to work on assignments and how many days they worked (see Figure 4), in some cases for students who start on the same day, those who end up working on fewer days tend to perform better in the course on average. One possible explanation for this is that high performing students require fewer days to complete assignments – prior work has shown that there are substantial

differences in the time it takes students to solve programming problems [11]. However, this also means that at least some students who exhibit the "ideal" work habits of both starting early and working on multiple days will nevertheless perform poorly in the exam. This suggests that there could be two cohorts of students exhibiting those patterns: students with good work habits, and students who are struggling and thus have to start early and have to work on multiple days. This means that interventions based purely on time management might miss some struggling students, and that interventions (e.g. additional support) should be aimed also at students who have already started work on assignments.

5.3 Changes in Earliness

We also observed that over the course, there is a change in earliness. That is, some students who start their work early in the first week(s) of the course start their work later in the subsequent course weeks. This does not hold for all students though, and some of the students are consistent in how they start their work early in the course. As starting to work later is linked with poorer course outcomes, additional insight into this behavior would be valuable.

In the course under study, the assignments are released weekly so that there are at least six days to work on the assignments. This means that this change in earliness cannot be attributed to overlapping deadlines within the course, and we can only speculate on the possible reasons. It is possible that this behavior is, for example, influenced by deadlines of other courses, by students adjusting their effort based on their expectations on the time that the assignments will take, or due to differences in the perceived value of the assignments.

5.4 Limitations

One limitation of our work is the lack of rich qualitative data that digs deeper into potential factors affecting students' use of time and their perception of their course assignments. For example, we did not account for students' workload such as how much other coursework they have in addition to their CS1 coursework (and e.g. the deadlines for other courses), how well they understood the topics, or the life-context of students outside of the university (e.g. part-time jobs that affect when they could do schoolwork); these could all influence how much time students could spend on their assignments and how many days they could work on assignments, and in turn, how well they perform on the course overall. Castro and Fisler found that even students' value judgements about the programming topics they are learning affects how well they perform on their assignments [3, 4]. Similarly, how students perceive the topics as beneficial to them could affect how much time they might allot for their work.

For the heatmap visualizations shown in Figure 4, we only included students who attended the final exam. This was done because students who did not attend the exam could have done so for a multitude of reasons: for example, it could be that they dropped out of the course due to struggling, but it could also be that they did not attend the final exam because they opted for the alternate exam (see Section 3.1.2) which we do not have data from. Thus, we decided to focus only on final exam attendees for that specific visualization. This, again, highlights the need for qualitative data that could help explain findings based on quantitative data.

The course from which our data was collected follows a "many small exercises" approach, which could have affected the results. For example, Denny et al. [7] found that students are likely to start smaller assignments earlier compared to more complex assignments. Additionally, there are many other course-specific factors (see Section 3.1 for details) that might influence the generalizability of our results.

6 CONCLUSIONS

In this work, we examined the relationship of students' time management in an introductory programming course and course outcomes. As a summary of our work, our answers to our research questions are as follows:

RQ1 *When do students start working on programming assignments and to what extent does their start time influence the number of days they take to work on their assignments?*

> **Findings:** We found that most students start working on course assignments relatively early, although a large proportion (about 40%) of students tend to have activity on the day of the deadline. Additionally, those who start on the last or second to last day before the deadline work on fewer days, but there are no big differences between those who start earlier than that.

RQ2. *How does when students start working on programming assignments and the number of days students work on programming assignments relate to their performance and effort in the course?*

> **Findings:** Our findings suggest that students who start earlier perform better in the course on average. However, we found that a majority of those who start just one day before the deadline still get the highest grade. Interestingly, we found that of those who start early, in some cases the students who spend more days working on the assignments perform more poorly compared to those early starters who work on fewer days.

Our results support prior work that found that students who start their coursework early perform better [10, 22, 23]. However, we found that some students who, based on the log data, have "ideal" work patterns (i.e. they both start work early and space their work over multiple days), still end up performing poorly in the course. This suggests that there is a need for interventions that also target students who have already started working and whose time management behavior seems optimal. One possible explanation for this is that based on our analysis, it is hard to differentiate between students who self-regulate effectively by starting work early and spacing their work, and those who start early and spend multiple days working because they are struggling.

Part of our future work involves collecting qualitative data, for example, through student interviews and surveys to get a richer picture of the factors underlying students' use of their time. These could augment the quantitative data we collect and further our understanding of the reasons behind different time management behaviors observed in this study.

REFERENCES

[1] T. Auvinen, N. Falkner, A. Hellas, P. Ihantola, V. Karavirta, and O. Seppälä. 2020. Relation of Individual Time Management Practices and Time Management of Teams. In *2020 IEEE Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/FIE44824.2020.9274203

[2] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the role of self-regulated learning on introductory programming performance. In *Proceedings of the first international workshop on Computing education research*. 81–86.

[3] Francisco Enrique Vicente Castro and Kathi Fisler. 2017. Designing a multi-faceted SOLO taxonomy to track program design skills through an entire course. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research (Koli Calling '17)*. Association for Computing Machinery, Koli, Finland, 10–19. https://doi.org/10.1145/3141880.3141891

[4] Francisco Enrique Vicente G. Castro. 2020. *Development of a Data-Grounded Theory of Program Design in HTDP*. Ph.D. Dissertation. Worcester Polytechnic Institute. https://digitalcommons.wpi.edu/etd-dissertations/595/

[5] Rubén Comas-Forgas and Jaume Sureda-Negre. 2010. Academic plagiarism: Explanatory factors from students' perspective. *Journal of Academic Ethics* 8, 3 (2010), 217–232.

[6] Peter F. Delaney, Peter P.J.L. Verkoeijen, and Arie Spirgel. 2010. Chapter 3 - Spacing and Testing Effects: A Deeply Critical, Lengthy, and At Times Discursive Review of the Literature. In *The Psychology of Learning and Motivation: Advances in Research and Theory*, Brian H. Ross (Ed.). Psychology of Learning and Motivation, Vol. 53. Academic Press, 63 – 147. https://doi.org/10.1016/S0079-7421(10)53003-2

[7] Paul Denny, Andrew Luxton-Reilly, Michelle Craig, and Andrew Petersen. 2018. Improving Complex Task Performance Using a Sequence of Simple Practice Tasks. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018)*. Association for Computing Machinery, New York, NY, USA, 4–9. https://doi.org/10.1145/3197091.3197141

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

[9] Hermann Ebbinghaus. 1885. *Über das gedächtnis: untersuchungen zur experimentellen psychologie*. Duncker & Humblot.

[10] 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.

[11] Fabian Fagerholm and Arto Hellas. 2020. On the Differences in Time That Students Take to Write Solutions to Programming Problems. In *2020 IEEE Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/FIE44824.2020.9274237

[12] Joseph R Ferrari and Catherine A Roster. 2018. Delaying disposing: examining the relationship between procrastination and clutter across generations. *Current Psychology* 37, 2 (2018), 426–431.

[13] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting academic performance: a systematic literature review. In *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*. 175–199.

[14] Douglas L Hintzman. 1974. Theoretical implications of the spacing effect. (1974).

[15] 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.

[16] 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 Conference on Global Computing Education*. 208–214.

[17] Ayaan M Kazerouni, Stephen H Edwards, T Simin Hall, and Clifford A Shaffer. 2017. DevEventTracker: Tracking development events to assess incremental development and procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. 104–109.

[18] Ayaan M Kazerouni, Stephen H Edwards, and Clifford A Shaffer. 2017. Quantifying incremental development practices and their relationship to procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 191–199.

[19] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. 2017. Comparison of time metrics in programming. In *Proceedings of the 2017 acm conference on international computing education research*. 200–208.

[20] Leo Leppänen, Juho Leinonen, and Arto Hellas. 2016. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 41–50.

[21] Krista Longi. 2016. *Exploring factors that affect performance on introductory programming courses*. Master's thesis.

[22] Therese H Macan, Comila Shahani, Robert L Dipboye, and Amanda P Phillips. 1990. College students' time management: Correlations with academic performance and stress. *Journal of educational psychology* 82, 4 (1990), 760.

[23] Joshua Martin, Stephen H Edwards, and Clifford A Shaffer. 2015. The effects of procrastination interventions on programming project success. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. 3–11.

[24] Ranjita Misra and Michelle McKean. 2000. College students' academic stress and its relation to their anxiety, time management, and leisure satisfaction. *American journal of Health studies* 16, 1 (2000), 41.

[25] Nick Parlante, Julie Zelenski, Daniel Zingaro, Kevin Wayne, Dave O'Hallaron, Joshua T Guerin, Stephen Davies, Zachary Kurmas, and Keen Debby. 2012. Nifty assignments. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 475–476.

[26] Paul R Pintrich et al. 1991. A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ). (1991).

[27] 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 Conference on International Computing Education Research*. 2–13.

[28] Jihae Shin and Adam M Grant. 2020. When Putting Work Off Pays Off: The Curvilinear Relationship Between Procrastination and Creativity. *Academy of Management Journal* (forthcoming) (2020).

[29] E. Soloway. 1986. Learning to Program = Learning to Construct Mechanisms and Explanations. *Commun. ACM* 29, 9 (Sept. 1986), 850–858. https://doi.org/10.1145/6592.6594

[30] 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.

[31] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 117–122.

[32] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2014. No tests required: comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 469–474.

[33] Melody Wiseheart, Annalise A D'Souza, and Jacey Chae. 2017. Lack of spacing effects during piano learning. *Plos one* 12, 8 (2017), e0182986.