The Impact of Programming Project Milestones on Procrastination, Project Outcomes, and Course Outcomes

A Quasi-Experimental Study in a Third-Year Data Structures Course

Clifford A. Shaffer
shaffer@vt.edu
Virginia Tech
Blacksburg, VA, USA

Ayaan M. Kazerouni*
ayaank@calpoly.edu
California Polytechnic State University
San Luis Obispo, CA, USA

ABSTRACT

When faced with a large and complex project for the first time, students face numerous self-regulatory challenges that they may be ill-equipped to overcome. These challenges can result in degraded project outcomes, as commonly observed in programming-intensive mid-level CS courses. We have previously found that success in these situations is associated with a disciplined personal software process. Procrastination is a prominent failure of self-regulation that can occur for a number of reasons, e.g., low expectancy of success, low perceived value of the task at hand, or decision-paralysis regarding how to begin when faced with a large task. It is pervasive, but may be addressed through targeted interventions. We draw on theory related to goal theory and problem-solving in engineering education to evaluate the value of explicit *project milestones* at curbing procrastination and its negative impacts on relatively long-running software projects. We conduct a quasi-experiment in which we study differences in project and course outcomes between students in a treatment (with milestones) and control group (without milestones). We found that students in the treatment group were more likely to finish their projects on time, produced projects with higher correctness, and finished the course with generally better outcomes. Within the treatment group, we found that students who completed more milestones saw better outcomes than those who completed fewer milestones. We found no differences in withdrawal or failure rates between the treatment and control groups. An end-of-term survey indicated that student perceptions of the milestones were overwhelmingly positive.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Software engineering education**; • **Software and its engineering** → *Software development process management*.

*Some work performed while at Virginia Tech.

KEYWORDS

procrastination, problem decomposition, software development, software project management

ACM Reference Format:
Clifford A. Shaffer and Ayaan M. Kazerouni. 2021. The Impact of Programming Project Milestones on Procrastination, Project Outcomes, and Course Outcomes: A Quasi-Experimental Study in a Third-Year Data Structures Course. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432356

1 INTRODUCTION

Procrastination is common among undergraduate Computer Science students. This "quintessential self-regulatory failure" [21] is often a result of students' well-documented difficulties with time management on software projects. It adversely affects their project outcomes [11] but can be addressed with interventions [14]. In this paper, we describe one such intervention and evaluate its effectiveness at curbing procrastination and its negative impacts.

Students in mid-level, programming-intensive CS courses experience self-regulatory difficulties while working on large or complex programming projects. Students may be attempting projects at a scale larger than what they have encountered in previous courses. The difficulties they face commonly manifest as procrastination and late or incomplete submissions. Though there are numerous causes for procrastination [21], we are primarily concerned with task-related procrastination, i.e., procrastination that tends to occur due to perceived properties of the task at hand. Examples are procrastination on tasks with distant outcomes [1], tasks for which students have low expectancy of success [4], and tasks that offer novices numerous junctures for decision-making [19].

We describe an intervention that exploits the ideas above to reduce the degree to which students procrastinate on large, complex, and relatively long-running software projects, and to ameliorate its negative effects. We provided *explicit project milestones*—sub-goals with intermediate deadlines that students were required to meet as they worked on the larger project. We applied this intervention in a post-CS2 Data Structures and Algorithms course (hereafter referred to as "CS3"), typically taken by juniors in our curriculum. We evaluated the impact of milestones on students' project outcomes using a previous instance of the course as a control group. Our results suggest that milestones had a positive impact on *timeliness*, *project correctness*, and *final course outcomes*.

The paper proceeds as follows: we begin by discussing related work in procrastination interventions and project decomposition in engineering education (§2). Then, we describe the project milestones (§3), define our research questions and study context (§4), and examine the impact that they had on timeliness, project correctness, and course outcomes (§5). Finally, we briefly present excerpts from an end-of-term survey (§6), assess threats to validity (§7), and close with a summary of our conclusions (§8).

2 RELATED WORK

Theoretical basis. Researchers have identified numerous causes and correlates for procrastination. While an individual's proclivity to procrastinate may be driven, to an extent, by personality traits [3], individuals in general are more likely to procrastinate on tasks with certain properties. *Goal theory* tells us that tasks whose outcomes are farther in the future are more likely to invite procrastination [1]. An individual's motivation to attempt a task tends to increase with their expectancy of successfully completing it [4].

Educators may be able to engineer situations in which task-related procrastination is less likely to occur. Ponton suggests that tasks with distant outcomes may still be addressed if they are broken down into sub-tasks that offer more proximal indicators of success [16]. The same holds for tasks that are perceived to be beyond one's ability: an individual's expectancy of success may be higher for a smaller sub-task than it is for the larger overarching task [4]. Further, an individual's self-efficacy regarding the larger task may itself increase with the completion of each successive sub-task. A common expert strategy is therefore to break down a complex engineering task into more manageable sub-tasks.

However, novices are often ill-equipped to perform this decomposition. It may be difficult for them to apply self-regulatory strategies without guidance [2, 20]. For example, Martin et al. [14] found that explicit email alerts were effective at keeping students on track to complete programming projects, even though they were only slightly tailored to a given individual's progress. In contrast, an "unguided" treatment was not. In this study, students wrote short reflections about their time management strategies used on each preceding project and their perceptions of how their strategies affected their behavior and performance. That intervention showed no changes in students' tendencies to start and finish their projects late, in comparison with a control group. Although they reported they appreciated the reflective exercise, students actually benefited more from explicit guidance provided by fairly generic email alerts.

Empirical studies. Good software process requires skills in self-regulation, which are difficult for novices to develop in any discipline. Falkner et al. report that many novice students found it difficult to reason about and improve their software processes, and so perform poorly on software development tasks [9]. They report that although CS students tended to improve most self-regulatory skills (e.g., design, testing) over the course of their undergraduate educations, *time management* remained a notable exception. Radermacher reports that this skill deficiency continues into their first jobs in industry, with hiring managers lamenting the inability of new hires to accurately estimate the time needed to complete software tasks [17]. This lack of time management ability often manifests as procrastination on software projects.

Numerous studies have found evidence for the adverse effects of procrastination on task outcomes (e.g., see Steel's meta-analysis [21]). Degraded outcomes due to procrastination have also been observed in software projects [8, 11]. Edwards et al. found that students tended to perform better on software projects when they made submissions earlier in the project timeline [8]. Similarly, Kazerouni et al. observed that when students worked on their projects earlier and more often, they produced projects with higher correctness and completed their projects earlier [11]. Both studies controlled for traits inherent to individual students, providing convincing evidence for the negative effect that procrastination has on software project outcomes.

Procrastination interventions. Procrastination can be addressed through targeted interventions. For example, Falkner et al. noted that novice students' inability to reason about their software development process may have hampered their ability to self-improve [9], and Tuckman noted that students developing self-regulatory abilities should be given the appropriate information needed to be aware of their progress toward completing a task [23]. As noted previously, Martin et al. sent periodic, adaptive emails to students regarding their progress on large software development projects in a CS3 course [14]. Students who received the email alerts had significantly earlier start times and significantly fewer late submissions than students in a control group.

Procrastination can be curbed with task-related strategies or interventions. For example, to counteract procrastination stemming from distant outcomes, one might decompose assigned tasks into sub-tasks with immediate indicators of success [16]. Prior efforts to guide students toward sub-goals in pursuit of a final solution have tended to be directed at novice programmers working on smaller programming projects (e.g., [12, 15]). Martin et al. studied intermediate CS students, and found their self-regulatory skills to also be lacking. This may be due to a lack of experience managing larger projects. We contend that this is a key contributor toward students' tendencies to delay working on projects. For example, students may not know how to start tackling a large project [19], or they may find it daunting [4], or the lack of frequent indicators of success may be demotivating [16], all known causes of procrastination [21].

3 PROJECT MILESTONES

To curb instances of procrastination on assigned programming projects with a 3-4 week lifecycle, we instituted *milestones*—specific increments of functionality that had to be completed by a given intermediate deadline. Milestones were satisfied by passing a specific subset of instructor-written reference tests on our automated assessment tool (Web-CAT [7]), and were worth at most 10% of the project grade. They were designed by the course instructor and generally were as follows:

- **Milestone 1**, due at week 1: Make an initial submission to the automated assessment system (in our case, Web-CAT [7])
- **Milestone 2**, due at week 1.5: Complete the data structure's *insertion* operation
- **Milestone 3**, due at week 2.5: Complete the data structure's *search* and *update* operations
- **Final Submission**, due at week 4: Complete the data structure's *removal* operation.

The ordering of milestones is partly due to dependencies between operations. For example, one cannot test a *search* or *update* operation without first implementing *insertion*, and *removal* operations typically depend on *search*.

Adhering to these milestones should, in theory, mitigate or eliminate many of the self-regulatory difficulties that novice software developers face when confronted with large programming tasks. For example, completing these milestones as described above would mean the *removal* operation is implemented last, after Milestone 3 and before the final submission. For many data structures, *removal* is the most difficult-to-implement operation, often involving complex changes in their structure (e.g., re-balancing a self-balancing tree, or re-ordering a binary search tree). Successively completing the other operations beforehand can help students increase their expectancy of success [16], making them less likely to procrastinate due to a lack of self-efficacy [4]. Additionally, we should observe reduced procrastination and fewer bad outcomes stemming from students' difficulties with decomposing a large engineering task [2, 19, 20], and from outcomes that are too far in the future to be valued in the present [1, 16, 22].

4 RESEARCH METHOD

4.1 Research Questions

We address the following research questions.

RQ1. Do project milestones improve students' timeliness on software projects? We have observed undesirably high rates of late submissions on assignments. We therefore examine the impact, if any, that project milestones had on late submission rates. We also examine whether the time of completion was related to the number of milestones that were successfully completed.

RQ2. Do milestones result in improvements to project correctness? In addition to late submissions, students are prone to turning in incomplete, incorrect, or inadequately tested solutions. We examine if milestones helped students turn in solutions with higher correctness (measured by instructor-written reference tests).

RQ3. Do milestones reduce the rates of incomplete or unsuccessful course outcomes? Finally, our CS3 course has a distressingly high rate of students dropping, failing, or withdrawing from the course (around 25–30% each semester). It is possible that incomplete or unsuccessful attempts at the course are a result of poor performance on early projects. We examine the impact that milestones had on overall course performance.

4.2 Study Context

We studied the work of students enrolled in a junior-level (third year) Data Structures and Algorithms course at Virginia Tech, a large public university in the United States. Students worked on three projects in Java in which they implemented one or more intermediate-to-advanced data structures (e.g., hash tables, quad trees, or self-balancing structures like B-trees) that are manipulated by commands in a text file with fairly simple syntax. Students were given about a month to work on each project. They were given no starter code and were free to design their own solutions from the ground up. However, they were also given significant guidance on what constituted a good design for the project.

Our data include submissions from two sections each in two semesters of the CS3 course. Sections in the Fall (Aug–Dec) 2013 semester were used as a control group, and sections in the Spring (Jan–Apr) 2016 semester were used as a treatment group (i.e., with milestones). All sections were taught by the same instructor (an author of this paper). More details can be found in Table 1.

To ensure originality of submitted work, projects were changed slightly between semesters. For the first project, one linear structure (a linked list) was replaced with another (a skip list). For the second project, a spatial data structure (a bin tree) was replaced with another that was similar (a quad tree). The third project in both semesters was a combination of the structures developed in the first two projects. Projects were similar between the treatment and control groups in terms of size and cyclomatic complexity.

Table 1: Control and treatment semesters. Designs are unbalanced because not all students attempted all projects, typically because they withdrew from the course.

	Control	**Treatment**
Semester	Fall 2013	Spring 2016
# Students	146	176
# Projects	3	3
# Submissions	406	481
LoC (median)	507.5	494
Cyc. Complexity (median)	218	234

5 ANALYSIS

5.1 RQ1: Timeliness

5.1.1 Impact on the Rate of Late Submissions. We examined the impact that project milestones had on the rate of late project submissions. We hypothesize that instituting milestones helped to reduce the rate of late submissions. We used Pearson's chi-squared test [10] to test this hypothesis. The test is used to determine if there is a relationship between two categorical variables. We used it to learn if there was a statistically significant difference in frequencies of on-time and late submissions between the control and treatment semesters. Categories and frequencies are in Table 2.

Table 2: Contingency table for on time and late submissions in the control and treatment semesters.

Semester	**On Time**	**Late**	**Total**
Treatment	422	59	481
Control	239	167	406
Total	661	226	887

We found a statistically significant difference in frequency of late submissions between the two semesters $\chi^2(DoF = 1, N = 887) = 95.11, p < 0.001$. The treatment semester saw a much lower rate of late submissions (12%) than the control semester (41%), suggesting that milestones helped students to complete projects on time.

5.1.2 *Impact on the Time of Project Completion.*

We assigned three milestones for each project, and not all students completed all milestones. To better understand the relationship between milestones and late work, we tested for a relationship between the *number* of milestones completed and the time of project completion (i.e., the time of the final graded submission).

We hypothesize that when students successfully complete more milestones—i.e., they turn in the required increment of functionality by the appropriate intermediate due date—they are likely to finish their projects earlier. We analysed submissions within the treatment group, since the control group had no milestones.

- **Dependent variable:** Project completion times in terms of the number of hours before (or after) the deadline (a negative number indicates a late submission)
- **Independent variable:** The number of milestones completed (0, 1, 2, or 3) as a categorical variable

We carried out a one-way analysis of variance (ANOVA) to determine if there were significant differences in completion times between submissions that satisfied different numbers of milestones.

Checking assumptions. A Shapiro-Wilk test [18] revealed that completion times were non-normal ($W = 0.80, p < 0.001$). Levene's test [5] revealed that the groups were homoscedastic, i.e., the samples had equal variances ($W = 0.32, p = 0.81$).

Based on the non-normality of the completion times and the homoscedasticity between milestone completion groups, we used the non-parametric Kruskal-Wallis ANOVA [13] to test for differences in completion times between milestone completion groups. The test revealed that were significant differences in completion times between groups ($H = 104.65, p < 0.001$).

Post-hoc analysis using Dunn's test [6] with a Bonferroni correction revealed significant pairwise differences in completion times. Significant differences in completion times were observed between submissions that had satisfied 0 and 2 milestones ($p = 0.006$), 0 and 3 milestones ($p < 0.001$), 1 and 2 milestones ($p < 0.001$), and 1 and 3 milestones ($p < 0.001$). Composite results—i.e., between the treatment and control groups, and within the treatment group—are depicted in Figure 1, where these differences are visually apparent.

In practical terms, when students satisfied 0 or 1 milestones, they tended to make submissions closer to (or even after) the deadlines. On the other hand, when they satisfied 2 or 3 milestones, they were likely to make their final submissions a day or more before the deadline. This suggests that project milestones may have helped students complete their projects earlier, possibly preventing late submissions. The cutoff of *2 or more* milestones is likely specific to our specific context. That is, Milestone 1 essentially only required students to "start working" on the project (§3).

5.2 RQ2: Project Correctness

While the original impetus for instituting milestones was to discourage procrastination, they offer other potential benefits due to encouraging project decomposition. Milestone submissions offer students an explicit strategy with which to approach projects, and students may feel more capable of tackling the resulting smaller sub-tasks [4, 16]. Problem decomposition is a common expert engineering strategy with which students often face difficulties [20]. The guided decomposition afforded by milestones may have helped

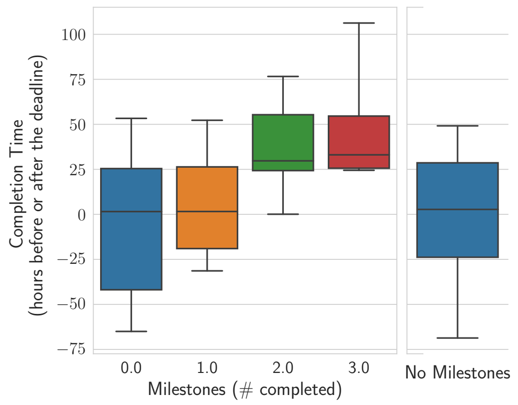

Figure 1: Completion times (in terms of hours before or after the deadline) with and without project milestones. Whiskers are the 10th and 90th percentiles. Outliers beyond these percentiles are omitted.

students to produce more correct project implementations. We note that the control group was encouraged to develop their projects incrementally in a similar way, so the differences are not simply due to additional knowledge on the part of the treatment subjects.

We hypothesize that milestones helped students to produce projects with higher correctness. To test this hypothesis, we used a t-test to check for differences in project correctness between submissions in the control semester and those in the treatment semester. We define the following variables:

- **Dependent variable:** Correctness, measured as the percentage of instructor-written reference tests passed by the submission
- **Independent variable:** Semester (*Control* or *Treatment*)

Checking assumptions. A Shapiro-Wilk test revealed that project correctness scores were not normally distributed ($W = 0.9, p < 0.001$). Levene's test revealed that the samples were not homoscedastic ($W = 18.01, p < 0.001$).

Based on the non-normality and heteroscedasticity of project scores, and the differences in sample sizes, we opted to use Welch's unequal variances t-test [24]. We found sufficient evidence to reject the null hypothesis that there is no difference in correctness scores between the control and treatment semesters ($t = 2.75, p = 0.006$).

This result suggests that instituting project milestones had a significant positive effect on the correctness of final submissions. Final submissions in the treatment semester had higher correctness scores ($\mu = 76\%, \sigma = 20\%$) than submissions in the control semester ($\mu = 72\%, \sigma = 21\%$).

Similar to the analyses in §5.1.2, we analyzed submissions within the treatment semester to understand the relationship between the project correctness and the *number* of milestones completed.

Checking assumptions. We know from earlier analyses that project scores were non-normal. Levene's test indicated that project scores were heteroscedastic between groups based on number of milestones completed ($W = 45, p < 0.001$).

Since project scores were heteroscedastic between groups, this analysis was done using a series of Welch's t-tests instead of the

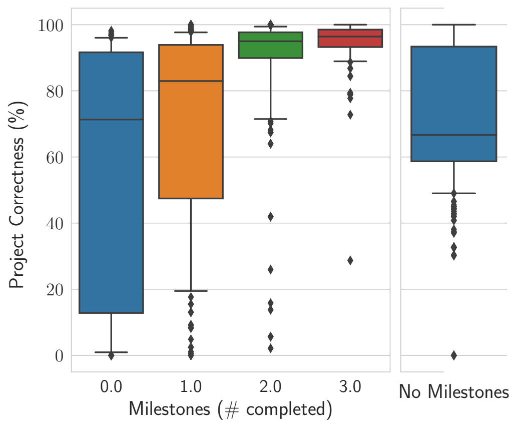

Figure 2: Correctness scores with and without milestones.

Kruskal-Wallis test, which assumes homoscedasticity. We used a Bonferroni correction to account for multiple comparisons. Significant pairwise differences in project correctness were observed between submissions that completed 0 and 2 milestones ($p < 0.001$), 0 and 3 milestones ($p < 0.001$), 1 and 2 milestones ($p < 0.001$), 1 and 3 milestones ($p < 0.001$), and 2 and 3 milestones ($p = 0.04$).

Figure 2 depicts composite results, i.e., between the control and treatment groups, and within the treatment group. Briefly put, students with milestones tended to produce projects with higher correctness than students without. Within the treatment group, reminiscent of results in §5.1.2, students tended to perform better when they completed more than 1 milestone.

5.3 RQ3: Course Outcomes

5.3.1 Impact on the Success of Course Attempts. Having seen that milestones had a positive impact on project outcomes (like timeliness and correctness), we now study impact on course outcomes, i.e., the pass, fail, and withdrawal rates in the course. Students at our institution tend to find the CS3 course challenging, and it is common to see 25–30% of students drop or withdraw from the course, or fail to achieve a grade that will let them progress on to subsequent courses. Success in the course is largely driven by success on the projects, and students may be withdrawing from the course based on early bad outcomes.

We tested for differences in frequencies of the following course outcomes between the control and treatment semesters:

- *Pass*: Student completed the course with a passing grade—they were able to progress in the CS degree
- *Fail*: Student completed with a failing grade—they would need to re-take the course to progress in the degree[1]
- *Withdraw:* Student withdrew from the course

We used Pearson's chi-squared test to test for independence between the course outcome variable (described above) and the semester (control or treatment). Frequencies are reported in Table 3.

The chi-squared test revealed no significant differences in course outcome frequencies between the two semesters $\chi^2(DoF = 2, N = 322) = 0.53, p = 0.77$. Milestones seemed not to have impacted the rate of unsuccessful attempts at the course (see Figure 3).

[1]Per §3, points allocated to the milestones had a negligible impact on the numerical final course grade.

Table 3: Contingency table for course outcomes in the control and treatment semesters.

Semester	Pass	Fail	Withdraw	Total
Treatment	136	20	20	176
Control	116	17	13	146
Total	252	37	33	322

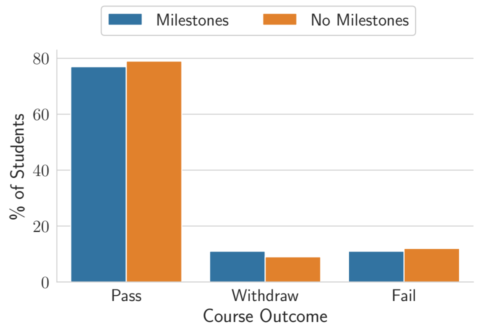

Figure 3: Milestones seem not to have affected course Pass, Fail, and Withdraw rates.

5.3.2 Impact on Final Course Grades. To further explore the impact of milestones on course outcomes, we examined differences in course grades between the control and treatment semesters. Differences in the frequencies of final grades may indicate the groups of students that benefit most from project milestones. That is, are the benefits we have observed localized to high- or low-performing students, or are they apparent for all students?

We used a chi-squared test to learn about differences in frequencies between occurrences of *A, B, C, D* and *F* final course grades between the control and treatment semesters (Table 4).

The chi-squared test indicated there was at least one significant difference in course grades between the control and treatment semesters $\chi^2(DoF = 4, N = 289) = 15.41, p = 0.004$. To explore which course grades had significantly different frequencies between semesters, we conducted post-hoc pairwise chi-squared tests, using Bonferroni corrections to account for the multiple comparisons. Post-hoc analysis revealed that that the only difference was between frequencies of *A* and *B* grades between semesters.

Table 4: Contingency table for course grades in the control and treatment semesters.

Semester	A	B	C	D	F	Total
Treatment	84	31	24	11	6	156
Control	45	51	21	9	7	133
Total	129	82	45	20	13	289

This result is depicted in Figure 4. More students received *A* grades in the semester with milestones (54%) than in the semester without milestones (34%). In contrast, fewer students received *B* grades in the semester with milestones (20%) than in the semester without milestones (38%). No other pairwise differences between nominal grade outcomes were observed.

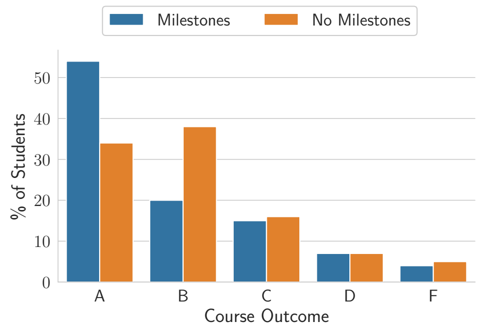

Figure 4: Final course grades with and without milestones.

This result suggests that the effects of milestones were more pronounced for students "in the middle" (about 20%) than for high- or low-performing students. Intuitively, many *B*-level students appear to have become *A*-level students. Unfortunately, students who performed poorly in the course (*C* grades or lower) continued to perform poorly. Other interventions or pedagogical methods must be explored to target the students who were not helped by milestones.

6 EXCERPTS FROM A SURVEY QUESTION

An informal end-of-term survey suggested that students generally appreciated having milestones. We have used project milestones nearly every semester since these results were observed (Spring 2016). During the next instance of the course (Fall 2016), we included the following question in an end-of-term survey:

> *How helpful did you find the Milestones in completing your programming projects on time? Please explain why you gave this response.*

Possible responses were *Not at all helpful*, *Not helpful*, *Neutral*, *Helpful* and *Very helpful*. The vast majority of students (75%) found milestones to be *Helpful* or *Very Helpful*. In their explanations, students mentioned that milestones helped them avoid procrastination:

> *I'm a procrastinator and they kept me on track*

They provided encouragement along the path to project completion:

> *The milestones reminded me to work and keep track of where I am... It felt nice seeing my progress on WebCAT*

Finally, they helped students break down the projects into logical and manageable sub-tasks:

> *It was really hard for me to figure out how to divide up projects, so not only did it divide up the projects for me, but it actually helped me learn how to divide up projects on my own.*

One student went so far as to say that milestones "likely prevented [them] from failing the class".

15% of students were *Neutral* about project milestones, saying that milestone deadlines were too close to each other, or that they did not need milestones to guide them. Only 10% of students found milestones to be *Not helpful* or *Not at all helpful*. They indicated that meeting milestones was often stressful, or that milestones interfered with their existing plans for the project.

7 THREATS TO VALIDITY

Internal validity. Differences in assigned projects could contribute to differences in project and course outcomes between the control and treatment groups. However, there were conceptual similarities between projects, as well as similarities in measurable proxies for "difficulty", like the size and complexity of submissions (see §4.2). These help to mitigate this threat.

External validity. Like any educational research, we are burdened by threats to the generalizability of this work. Our large sample sizes may have helped to mitigate this threat. The key finding that milestones helped to shift a significant number of B grades to A grades also holds up well over time. The grade distribution shown for the control group is similar to that seen in years prior, while the grade distribution for the treatment group has continued in subsequent semesters with the continued use of milestones.

Conclusion validity. It should be noted that this is a quasi-experimental study, i.e., students were not randomly assigned to the treatment and control groups. However, our experimental design and sample size makes us reasonably confident of a causal relationship between project milestones and the observed positive impacts on course and project outcomes.

8 CONCLUSION

Though we cannot state with certainty *why* project milestones were successful, related work presented in §2 provides solid theoretical grounding for our results. Further, students' perceptions of how exactly milestones helped them are in agreement with the well-known benefits of setting sub-goals in engineering education. This increases our confidence in our findings that milestones curbed procrastination and improved project and course outcomes.

A possible criticism of this work is that our milestone requirement takes agency away from students on important points related to time management and development, and they therefore do not learn these project management skills (because "we did it for them"). Our underlying pedagogical view is that project management is a skill. Skills are normally taught through guided practice, e.g., learning to swim. Students coming to our course have at least two semesters of programming experience, but relatively little experience in implementing larger projects. At best, they will have experienced one major project before, usually the last project of our CS2 course. This is a core competency that our CS3 course is meant to teach. We believe that milestones not only work to improve project outcomes, but they provide meaningful long-term results in two ways: (1) they force students to practice a successful approach to large project implementation, and (2) they train students to experience success at developing large projects. Even if students abandon the principles that they practiced in their next course, they would have prior experience of success to guide their evaluation of future projects. This represents important future work: to examine the long-term effects of milestones as a practice mechanism.

ACKNOWLEDGMENTS

We thank the anonymous peer reviewers for their feedback and Dr. J. Davis for critiquing drafts of this paper.

REFERENCES

[1] George Ainslie. 1975. Specious reward: a behavioral theory of impulsiveness and impulse control. *Psychological bulletin* 82, 4 (1975), 463. https://doi.org/10.1037/h0076860

[2] T.N. Arvanitis, M.J. Todd, A.J. Gibb, and E. Orihashi. 2001. Understanding students' problem-solving performance in the context of programming-in-the-small: an ethnographic field study. In *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193)*. IEEE, Reno, NV, USA, F1D–20–3. https://doi.org/10.1109/FIE.2001.963676

[3] Richard D. Arvey, Maria Rotundo, Wendy Johnson, Zhen Zhang, and Matt McGue. 2006. The determinants of leadership role occupancy: Genetic and personality factors. *The Leadership Quarterly* 17, 1 (2006), 1 – 20. https://doi.org/10.1016/j.leaqua.2005.10.009

[4] Albert Bandura. 1997. *Self-Efficacy: The Exercise of Control*. Macmillan.

[5] Morton B. Brown and Alan B. Forsythe. 1974. Robust Tests for the Equality of Variances. *J. Amer. Statist. Assoc.* 69, 346 (1974), 364–367. https://doi.org/10.1080/01621459.1974.10482955 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10482955

[6] Olive Jean Dunn. 1964. Multiple Comparisons Using Rank Sums. *Technometrics* 6, 3 (1964), 241–252. https://doi.org/10.1080/00401706.1964.10490181 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/00401706.1964.10490181

[7] Stephen H. Edwards. 2004. Using Software Testing to Move Students from Trial-and-Error to Reflection-in-Action. *SIGCSE Bull.* 36, 1 (March 2004), 26–30. https://doi.org/10.1145/1028174.971312

[8] 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 (ICER '09)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/1584322.1584325

[9] Katrina Falkner, Rebecca Vivian, and Nickolas J. G. Falkner. 2013. Neo-Piagetian Forms of Reasoning in Software Development Process Construction. In *Proceedings of the 2013 Learning and Teaching in Computing and Engineering (LATICE '13)*. IEEE Computer Society, USA, 31–38. https://doi.org/10.1109/LaTiCE.2013.23

[10] Karl Pearson F.R.S. 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175. https://doi.org/10.1080/14786440009463897 arXiv:https://doi.org/10.1080/14786440009463897

[11] 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* (Tacoma, Washington, USA) *(ICER '17)*. ACM, New York, NY, USA, 191–199. https://doi.org/10.1145/3105726.3106180

[12] Aaron Keen and Kurt Mammen. 2015. Program Decomposition and Complexity in CS1. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15*. ACM Press, Kansas City, Missouri, USA, 48–53. https://doi.org/10.1145/2676723.2677219

[13] William H. Kruskal and W. Allen Wallis. 1952. Use of Ranks in One-Criterion Variance Analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621. https://doi.org/10.1080/01621459.1952.10483441 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1952.10483441

[14] Joshua Martin, Stephen H. Edwards, and Clfford 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* (Omaha, Nebraska, USA) *(ICER '15)*. ACM, New York, NY, USA, 3–11. https://doi.org/10.1145/2787622.2787730

[15] Briana B. Morrison, Lauren E. Margulieux, and Mark Guzdial. 2015. Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. In *Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER '15*. ACM Press, Omaha, Nebraska, USA, 21–29. https://doi.org/10.1145/2787622.2787733

[16] Michael K. Ponton, Julie Horine Edmister, Lawrence S. Ukeiley, and John M. Seiner. 2001. Understanding the Role of Self-Efficacy in Engineering Education. *Journal of Engineering Education* 90, 2 (April 2001), 247–251. https://doi.org/10.1002/j.2168-9830.2001.tb00599.x

[17] Alex Radermacher, Gursimran Walia, and Dean Knudson. 2014. Investigating the Skill Gap between Graduating Students and Industry Expectations. In *Companion Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE Companion 2014)*. Association for Computing Machinery, New York, NY, USA, 291–300. https://doi.org/10.1145/2591062.2591159

[18] Samuel S. Shapiro and Martin B. Wilk. 1965. An analysis of variance test for normality (complete samples)†. *Biometrika* 52, 3-4 (12 1965), 591–611. https://doi.org/10.1093/biomet/52.3-4.591 arXiv:https://academic.oup.com/biomet/article-pdf/52/3-4/591/962907/52-3-4-591.pdf

[19] Maury Silver. 1974. Procrastination. *Centerpoint* (1974).

[20] Ting Song and Kurt Becker. 2014. Expert vs. novice: Problem decomposition/recomposition in engineering design. In *2014 International Conference on Interactive Collaborative Learning (ICL)*. IEEE, Dubai, United Arab Emirates, 181–190. https://doi.org/10.1109/ICL.2014.7017768

[21] Piers Steel. 2007. The nature of procrastination: A meta-analytic and theoretical review of quintessential self-regulatory failure. *Psychological Bulletin* 133, 1 (2007), 65–94. https://doi.org/10.1037/0033-2909.133.1.65

[22] Jennifer Stock and Daniel Cervone. 1990. Proximal goal-setting and self-regulatory processes. *Cognitive Therapy and Research* 14, 5 (Oct. 1990), 483–498. https://doi.org/10.1007/BF01172969

[23] Bruce W. Tuckman. 1991. The Development and Concurrent Validity of the Procrastination Scale. *Educational and Psychological Measurement* 51, 2 (June 1991), 473–480. https://doi.org/10.1177/0013164491512022

[24] B. L. Welch. 1947. The Generalization of 'Student's' Problem When Several Different Population Variances are Involved. *Biometrika* 34, 1-2 (01 1947), 28–35. https://doi.org/10.1093/biomet/34.1-2.28 arXiv:https://academic.oup.com/biomet/article-pdf/34/1-2/28/553093/34-1-2-28.pdf