AlgoScrum: Improving an Algorithms Course with Scrumage*

Scott Spurlock and Shannon Duvall
Computer Science
Elon University
Elon, NC 27302
{sspurlock,sduvall2}@elon.edu

Abstract

We present the design of an Algorithms Analysis course, based on the recently developed Scrumage approach, which allows students to work in cohorts if they choose and also allows each student to choose for themselves how they spend class time. We describe the course structure as well as the results from a survey to assess learning attitude outcomes. We show that the Scrumage method resulted in students taking more responsibility for their own learning and having improved impressions of the course and the course material.

1 Introduction

Algorithm Analysis is at the core of computer science [13]. At our institution, the anecdotal reputation of the course is somewhat negative, with end-of-term surveys from prior semesters surfacing student comments such as "material is miserable," "so much information," and "very confusing." Our observations are that students frequently struggle to engage with the material and, worse, often display a lack of understanding of how to improve, as well as a degree of passivity in their own learning.

In order to address these issues, we re-worked the course to follow the recently developed Scrumage pedagogical approach [6]. In Scrumage (SCRUM for AGile Education), course content is divided into short (2-3 week) units called *sprints*, and students work in self-organizing teams to complete requirements (e.g., problem sets and programming assignments). A key component of Scrumage is that students have the freedom to choose multiple different learning approaches. Thus, one student may choose a traditional in-class lecture, while another may prefer a flipped model, watching videos outside of class instead. Students are provided with a variety of resources each sprint (textbook readings, slides, videos, sample problems, etc.) to allow them to dynamically choose how to learn the material.

We were interested to apply the Scrumage model to our Algorithms course with an eye toward helping students take more responsibility for their own learning and allowing them to have more choice in how they learn. With this idea in mind, we redesigned the course to include five sprints, each with its own set of assignments and a quiz at the end. Students had the freedom to choose a team (or to work individually) each sprint and to decide how to use the provided resources as they saw fit.

We have taught our Algorithms course using Scrumage for the past three offerings, observing each time improving student attitudes and evidence of metalearning, as students discover how they learn best. As instructors, we have found the experience of adopting Scrumage to have made the teaching process more enjoyable as students have transitioned psychologically from a model where a teacher pushes the material at them, to a model where they are empowered to make decisions about their own learning experience.

2 Related Work

As might be expected given the central importance of the topic, a variety of work has been published over the years related to teaching Algorithms to undergraduate students, including various pedagogical alterations to improve student learning. For example, one recent paper describes modifying a traditional course to focus on group-based problem solving [3], while another adapts a course to use an interactive e-book [9]. Including peer assessment in one algorithms course helped enhance students' critical thinking skills [5]. Other work looks at incorporating card games [10], puzzles [12], or programming competitions [8] to increase student interest.

The variety of approaches in the literature suggests that there is no single best pedagogy for teaching Algorithms. It seems clear from the scholarship of teaching and learning more broadly that active learning approaches are effective [11], but there appears to be no consensus on the effectiveness of particular

techniques such as lecture [4], gamification [1], or a flipped model [2]. This observation informs the development of Scrumage, which aims to allow for all of these approaches to be available simultaneously to students within a single classroom [6]. It is inspired by Scrum, a widely adopted project management technique that emphasizes lean processes (no busy-work), autonomous teams (no dictated decisions from management), and a fast feedback loop on both process and product (no static plans) [16]. Scrumage has previously been applied to a Discrete Mathematics course with a resulting improvement in student attitudes [7], but no work exists on using Scrumage for a higher-level course like Algorithm Analysis. In the following section, we describe the design of a Scrumage-based Algorithms course as well as a survey constructed to assess how student attitudes about learning changed from the start to the end of the semester.

3 Methodology

We have taught Algorithm Analysis at our institution many times in the past following a traditional, lecture-based approach; we have used the Scrumage approach in each of the last three offerings, observing each time similar improvements compared to our prior traditional approach. The authors have individually implemented Scrumage with slight differences in content, assignments, and incentives, but with the same core principles. In the following sections, we describe the most recent offering.

3.1 Course Structure

In the Scrumage approach, a course is divided into a series of units called sprints, each with its own set of topics and requirements. Each sprint begins with team assignment (after the first sprint, students have primary input into team formation) and the distribution of available resources (videos, readings, slides, etc.) and requirements (work to be completed by the end of the sprint). We provide a form for teams to make requests for how class time should be spent each day of the sprint. For example, a student might request the instructor to work through example problems similar to one of the requirements on a given day or play a review game prior to a quiz. Students are required to come to class each day to meet with their team at the start of class and to complete a check-in problem - a short (5-minute), lightly graded quiz designed to help students better understand how well they are progressing. After these required activities, students have the liberty to stay for the remainder of class or not, depending on whether the planned activities are helpful to them or not. On many days there is an optional lecture, followed by in-class work time when students can make progress on their requirements while the instructor

Sprint	Topics	Requirements
1. Fundamentals I	Basics	PS1 - basic analysis
(3 weeks)	Analysis	PA1 - unique elements
	Sorting	Quiz 1
	Divide & conquer	
	Master method	
2. Fundamentals II	Recurrence relations	PS2 - more analysis
(3 weeks)	Quicksort	PA2 - k largest
	Lower bounds	Quiz 2
	Linear sorting	
3. Data Structures	Data structures	PS3 - heaps and trees
(3 weeks)	Binary search trees	Quiz 3
	Hash tables	
4. Graphs	Graphs, DFS, BFS	PS4 - graphs and greedy
(3 weeks)	Dijkstra's Algorithm	PA3 - fastest route
	MST	Quiz 4
	Huffman trees	
5. Adv. Techniques	Dynamic programming	PS5 - DP & NPC
(2 weeks)	NP-Completeness	Final exam

Table 1: The course is divided into 5 sprints, each focused on a subset of topics and with defined requirements to be completed. The requirements are either team-based problem sets (PS) or individual programming assignments (PA).

circulates providing help as requested. The last day of each sprint culminates in a quiz.

Table 1 shows a breakout of the topics and assignments for each of the five sprints. In particular, there are problem sets covering the more theoretical elements of algorithm design and analysis (15% of the total points), as well as more practical programming assignments (20% of the total points). The first four sprints end with a quiz (32% of the total points). Material from the fifth sprint is included on the (cumulative) final exam (23% of the total points). A small number of points (5%) is awarded for learning management activities, such as attending team meetings, making and following through on plans, and completing a retrospective survey at the end of each sprint. Finally, the daily feedback check-ins comprise a small number of points (5%) so students complete them thoughtfully.

3.2 Survey Development

To help identify changes in student attitudes over the course of the semester, we created a survey to be administered at the start and end of the course. The survey included 30 Likert-scale (1 - 7) questions. Of these, six related to student learning preference, such as "Small group discussion is an effective approach for me when I am learning new material." Also included were 24 questions focusing on student learning attitudes taken from the "Motivated Strategies for Learning Questionnaire" [15] from 4 categories: Effort Regulation, Metacognitive Self-Regulation, Help Seeking, and Control of Learning Beliefs. The surveys also included several free-text fields, such as "What are your impressions of the topic of Algorithm Analysis?" In the following section, we describe the survey results and our observations of student learning and attitudes.

4 Results and Discussion

The most recent course offering included 31 students across two sections. The majority of students were in their 3rd or 4th year in the program, with a few students in their 2nd year. We administered the survey the week prior to the start of the course, which established a baseline for our analysis, as well as providing input to initial team formation, where students providing similar survey responses were grouped together.

Of the 4 categories, the largest change in average student score was in the Control of Learning Beliefs category, which focuses on the extent to which students feel that they are able to learn the course material and are responsible for their own learning outcomes. While the sample size is small, this change was found to be statistically significant ($p = 0.05$) using the Wilcoxon signed rank test. This outcome suggests that we met one of our key goals: to help students take responsibility for their own learning. Figure 1 shows the distribution of student responses on each category of student attitude questions using kernel density estimation (KDE) [14] for the pre- and post-surveys.

In particular, the student attitudes questions with the largest absolute change over the semester were, "It is my own fault if I don't learn the material in this course," which is part of the Control of Learning Beliefs category, and "Even if I have trouble learning the material in this class, I try to do the work on my own, without help from anyone," which is part of the Help Seeking category. We observed an average decrease in the Help Seeking category score, which we attribute to students being more inclined to make use of the provided resources rather than asking the instructor for help. While this outcome could indicate more independence in learning, we did not observe a decrease in student questions in class or by email, and attendance in office hours remained strong throughout the course. Other responses that showed significant

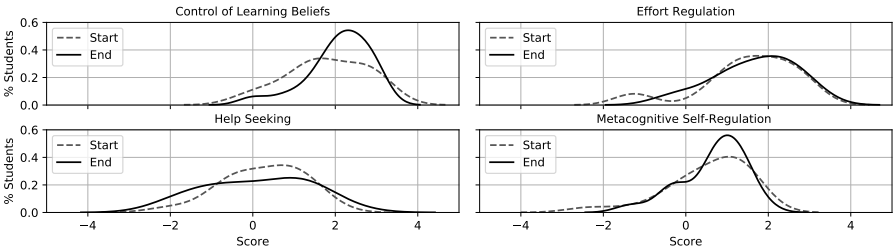

Figure 1: Kernel Density Estimate (KDE) distributions of average scores across categories from the learning attitudes survey questions show how student responses changed from the start to the end of the semester.

increases related to learning preferences for working in small groups and for "jumping straight into problem-solving and looking up relevant information along the way" as opposed to a more structured introduction to material.

4.1 Text responses

In addition to Likert-scale questions, students responded to several free-text prompts. The following sections focus on changes in student impression of the topic, their feelings about ownership of their own learning, and their metalearning about their own individual approach to the course.

4.1.1 Student Impressions

One survey question related to student perceptions of the topic of algorithm analysis. Running the Vader sentiment intensity analyzer shows an improvement in student sentiment from pre- to post-survey. In particular, the compound sentiment score, which varies from -1 for extremely negative to +1 for extremely positive, increased from 0.1254 to 0.4543, over the semester. For additional insight, we coded the responses based on keywords related to four categories: difficult, interesting, useful, and enjoyable. The following example comments are typical for each category:

- Difficult: "Algorithm analysis was very difficult... but I learned a lot from the class. "
- Interesting: "Tough but interesting. I like the more academic lens of computer science that we have in this class."
- Useful: "It seemed to be very important and mostly a way of thinking. It helped me think of new ways to solve problems and approach

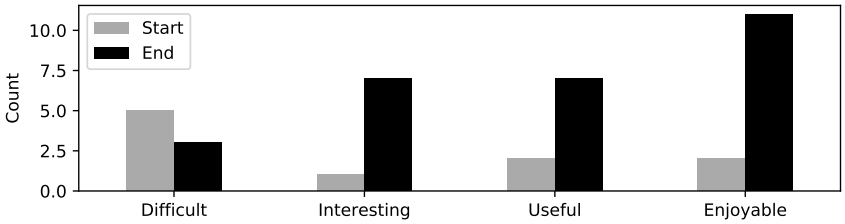

Figure 2: Counts of sentiment tags relating to student impressions of the topic of algorithms on a pre- and post-test.

programming from a different perspective."

- Enjoyable: "I enjoyed it. I liked going beyond writing functional code and evaluating what made code efficient/fast/generally 'good'."

Figure 2 shows how these counts changed over the course of the semester, with students indicating increased positive impressions and decreased negative impressions. (Note that a response could be tagged with multiple categories.)

4.1.2 Student Ownership of Learning

One interesting observation is that, while in-class lectures were explicitly labeled as optional, with students invited to leave if they preferred a different learning approach, in practice, almost all students stayed every day for lecture. However, they noted in surveys that they appreciated the freedom to choose, e.g., "every day I made the choice to stay for the lectures" and "... the optional lectures helped because I always stayed to advance my learning." We hypothesize that the psychological impact of "opting in" is a key part of the success of the Scrumage approach. One survey question related specifically to students' feelings about responsibility for their own learning. Multiple responses suggest that the Scrumage approach was effective at promoting student ownership:

- "I think this learning approach was successful for getting me to take ownership of learning material. It allowed me to be able to study at my own pace ... Compared to my other classes I would honestly rate this one the highest in terms of how much I've been motivated to understand the concepts and it's the class I feel I've learned the most in."
- "This freedom to decide between videos, slides, readings, and lectures, not only made class more meaningful, but also helped dampen the feeling that showing up to class was a chore (ex. if you felt comfortable with the topic, the lecture may not have been necessary to stay for)."

- "... [The] sprint style was my favorite class organization I have seen... The division into smaller topics helped make the course load feel more manageable, and allowed additional emphasis to be put on each topic. Further by having group and individual work for each sprint, I was able to improve my teamwork skills, and also ask questions to help work through problems and facilitate my own discussion and learning."

4.1.3 Metalearning

One survey question asked students to indicate their strategy for using the available resources to learn the material. Gratifyingly, many students reported progress in learning about their own learning:

- "I learned over the semester that my best way to learn was to watch the videos and look over the slides before a class. This way, I could ask questions during the lecture, and reinforce what I had taught myself."
- "I generally went over lecture slides first, then if I still had trouble understanding material I went to the book, and if I still wasn't sure I used the videos. I tried to understand the material first before going into solving problems so if I got stuck I would know where to refer to for answers."
- "I primarily learned from jumping into the problems and trying to piece it together with videos textbook and slide. If all else failed I went straight to office hours."
- "... My approach changed slightly over time as I realized the pre class prep is what helped me the most for truly understanding the topics."

Coding the free text responses indicates that lectures were the most popular learning approach, closely followed by videos (Figure 3). Of note, most students mentioned preferring multiple learning modalities.

5 Conclusion and Future Work

Our results suggest that the Scrumage teaching approach can be effectively applied to an Algorithms course and resulted in students taking more ownership over their learning, discovering how they personally learn best, and having better impressions of the course and the topic of Algorithm Analysis.

While we saw a marked increase in attitudes, grade outcomes from the course were similar to semesters that followed a traditional pedagogical approach. However, because a number of factors change from one course offering to another, no clear conclusions can be drawn from this outcome. We look

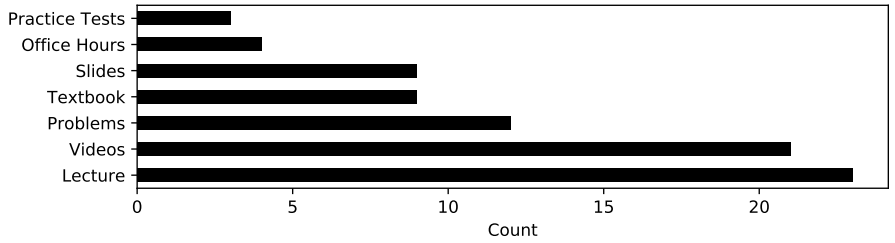

Figure 3: Counts of which learning approaches were mentioned in student responses to a survey question about how students chose to learn material.

forward to further studying the effect on content learning as well as the effect on instructor experience in the future.

Finally, we note that, from the instructor perspective, Scrumage resulted in a more enjoyable teaching experience. Students appeared more motivated and better prepared for class, often asking better questions that suggested they had already attempted to solve homework problems on the topic of discussion. Our experience has been that less time is needed for spoon-feeding students the basics, and more time is spent on applications of techniques and drawing connections between different ideas.

References

[1] Azita Iliya Abdul Jabbar and Patrick Felicia. Gameplay engagement and learning in game-based learning: A systematic review. *Review of educational research*, 85(4):740–779, 2015.

[2] Jacob Lowell Bishop, Matthew A Verleger, et al. The flipped classroom: A survey of the research. In *ASEE national conference proceedings, Atlanta, GA*, volume 30, pages 1–18, 2013.

[3] Florent Bouchez-Tichadou. Problem solving to teach advanced algorithms in heterogeneous groups. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, page 200–205. ACM, 2018.

[4] Mary Burgan. In defense of lecturing. *Change: The Magazine of Higher Learning*, 38(6):30–34, 2006.

[5] Donald Chinn. Peer assessment in the algorithms course. *SIGCSE Bull.*, 37(3):69–73, June 2005.

[6] Shannon Duvall, Dugald Ralph Hutchings, and Robert C Duvall. Scrumage: A method for incorporating multiple, simultaneous pedagogical styles in the classroom. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 928–933, 2018.

[7] Shannon Duvall, Duke Hutchings, and Michele Kleckner. Changing perceptions of discrete mathematics through scrum-based course management practices. *Journal of Computing Sciences in Colleges*, 33(2):182–189, 2017.

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

[9] Tommy Färnqvist, Fredrik Heintz, Patrick Lambrix, Linda Mannila, and Chunyan Wang. Supporting active learning by introducing an interactive teaching tool in a data structures and algorithms course. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, page 663–668. ACM, 2016.

[10] Lasse Hakulinen. Card games for teaching data structures and algorithms. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, page 120–121. ACM, 2011.

[11] Alison King. From sage on the stage to guide on the side. *College teaching*, 41(1):30–35, 1993.

[12] Anany Levitin and Mary-Angela Papalaskari. Using puzzles in teaching algorithms. *SIGCSE Bull.*, 34(1):292–296, February 2002.

[13] ACM Joint Task Force on Computing Curricula. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science, 2013.

[14] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[15] P. R. Pintrich, D. Smith, T. Garcia, and W. McKeachie. A manual for the use of the motivated strategies for learning questionnaire (mslq). 1991.

[16] Jeff Sutherland and JJ Sutherland. *Scrum: the art of doing twice the work in half the time*. Currency, 2014.