Increased Performance via Supplemental Instruction and Technology in Technical Computing

Dr. Nathan L Anderson, California State University, Chico

Dr. Nathan L. Anderson is an Assistant Professor in the Department of Mechanical and Mechatronic Engineering and Sustainable Manufacturing at California State University Chico. He engages in multiple research projects spanning computational materials science to educational pedagogy. Prior to joining academia, he worked in the semiconductor manufacturing industry for KLA Corporation. Before industry, he spent time at Sandia National Laboratories. He earned his Ph.D. in Materials Science and Engineering from Purdue University and his B.S. in Materials Engineering from San Jose State University.

Increased Performance via Supplemental Instruction and Technology in Technical Computing

Nathan L. Anderson
Department of Mechanical and Mechatronic Engineering and Sustainable Manufacturing, California State University Chico

Abstract

The introduction of programming to multiple engineering disciplines within a large classroom environment presents many challenges. It is quite well established that some sort of hands-on laboratory or activity, providing practice for the student, is essential for successful learning and retention of programming. Feedback time during these sessions becomes more limited as the number of students increases, hence supplemental instruction (SI) can be utilized to increase feedback and student interactions. Here, we demonstrate how the implementation of SI, as developed by UMKC, in combination with tablet based demonstrations and hand-written/program-specific examples are effectively used to improve student grades and course evaluations. Weekly SI sessions were developed to reiterate key concepts from the lab for that week and also provided students with a peer-friendly environment where they could engage in questions/discussion without the presence of the course instructor. Grade improvement is seen by nearly eliminating the failure rate and a statistically significant shift in the overall distribution upward from previous offerings. Improvement of student evaluations are also highlighted indicating positive responses to teaching methodology as well as supplemental instruction.

Introduction

This paper is based on evidence-based practice. It is well established [1]-[2] that introductory programing, or technical computing, is a difficult topic for students to grasp during their initial exposure. New nomenclature, use of new software, and structural elements of programming (controls, loops, algorithms) are just some of the hurdles that need to be overcome in an introductory course. While it has been demonstrated [2] that a hands-on component, such as a weekly laboratory or activity, helps to improve performance, there still exists both educational and communication gaps between the student and instructor in an instructor only based environment. Some examples of these gaps include inability to break down complex tasks into smaller subsets, lack of peer instruction, intimidation associated with addressing the instructor directly, and lack of practice or example problems externally generated (outside the course instructor). Supplemental Instruction (SI) is a concept introduced by University of Missouri-

Kansas City in 1973 [4] and is particularly well suited to address these gaps for an introductory programming course.

The primary components of SI include: SI leaders who attend the regular scheduled lectures and complete all course assignments, multiple weekly SI leader led teaching sessions, evaluation of sessions by SI supervisors for feedback and improvement, weekly planning and coordination of session content between SI-leader and course instructor. Prior to the class start date, SI leaders receive training on session preparation and teaching pedagogy, and work with SI supervisors and faculty to continually monitor and modify session content. SI was developed around a combination of learning theories [5], cognitive development principles [6], societal interdependence principles [7], and interpretive principles [8]. Specifically, the four aforementioned gaps applicable to technical computing can be filled by the implementation of SI.

Methodology

The course targeted for this study is titled “Introduction to Technical Computing”. It is a sophomore/junior level course intended to introduce programming in Matlab, VBA, and Python. In addition to SI, this course lecture format was also redesigned with the goal of better overcoming the aforementioned hurdles. The prior lecture format used a combination of Powerpoint slides, with code already written in the slides, and opening the actual program being taught to show examples as well. In the redesigned course the Powerpoints were eliminated and replaced with a tablet where the instructor is able to deliver hand-written examples that can then be demonstrated in the program being taught, for the purposes of this paper this lecture redesign is referred to as technology. This technology also involved focusing on practical examples rather than abstract theory (i.e. use a for loop to calculate a sum vs. the general structure of a for loop is…). It is common knowledge that by starting with a blank board (or in this case tablet screen), and populating in real-time that students are much more likely to replicate the instructor’s actions. Actually writing the code, before looking at it demonstrated in a program, significantly helps understand structural elements of programming [2].

When tasked with writing a program to solve a particular problem there are typically multiple components, and students struggle breaking down these components into smaller more manageable subsets. Often, due to time constraints, the instructor does this breakdown during the traditional lecture at a pace that is difficult for first-exposure students to grasp. Combining this with the large class populations, limited lectures, and limited office hours that typically accompany an introductory programming course, there are multiple contributing factors toward the student’s inability to get one-on-one or peer led instruction on this breakdown process. In the case of this class, the SI leader is trained to implement sessions that focus solely on problem

breakdown. An example session focused on the flight of a rocket and specifically analyzing the duration a rocket burned fuel over a specific height. This is at first overwhelming for an introductory student, but the SI leader shows how to calculate various subsets such as initial values, total flight path for the rocket, height for total burn time, and finally duration of burning above a specified height. Ideally, the students participating in the session are responsive and the SI leader offers positive reinforcement to guide them toward applicable subsets.

Another major benefit of SI is having a peer, the SI leader, provide an opportunity for collaborative enagement during the sessions [9]-[10]. This has a multifold advantage with regards to the gaps mentioned above. It offers an alternative route for students questions and participation outside of directly interacting with the course instructor, thus greatly alleviating any anxiety or intimidation a student may have. Also, while the high-level session topics are discussed between instructors and SI leaders, the specific details of a session (such as example problems) are up to the SI leaders themselves. Therefore, the final gap aforementioned is filled via any additional problems developed by the SI leaders and covered in their sessions. SI leaders are informed not to answer questions they are uncertain of and refer those questions to the instructor. This serves as a failsafe for misinformation which leads to distrust and ultimately unfaithfulness in SI as whole. Data collection and analysis is discussed in the results.

Results

Two versions of an introduction to technical computing class, with the same content, were taught to evaluate student performance. The classes were taught sequentially, one semester after another, and were taught by the same instructor to avoid any confounding variation. One in the traditional matter with no technology/SI implementation as well as a second redesigned version with the technology/SI implementation. In the redesigned version the SI leader went through both pedagogy and problem break-down trainings, led two weekly SI sessions, and created all their own content for the sessions. Assessment measures such as homework, quizzes, and exams written to be equivalent from a conceptual and difficulty standpoint. One of the key indicators of success for SI is improvement of the DFW (students receiving a D, F, or W) rate [4]. Figure 1 shows a histogram of grade distributions from both the pre-redesign (non-SI) and redesign (SI) versions of the course.

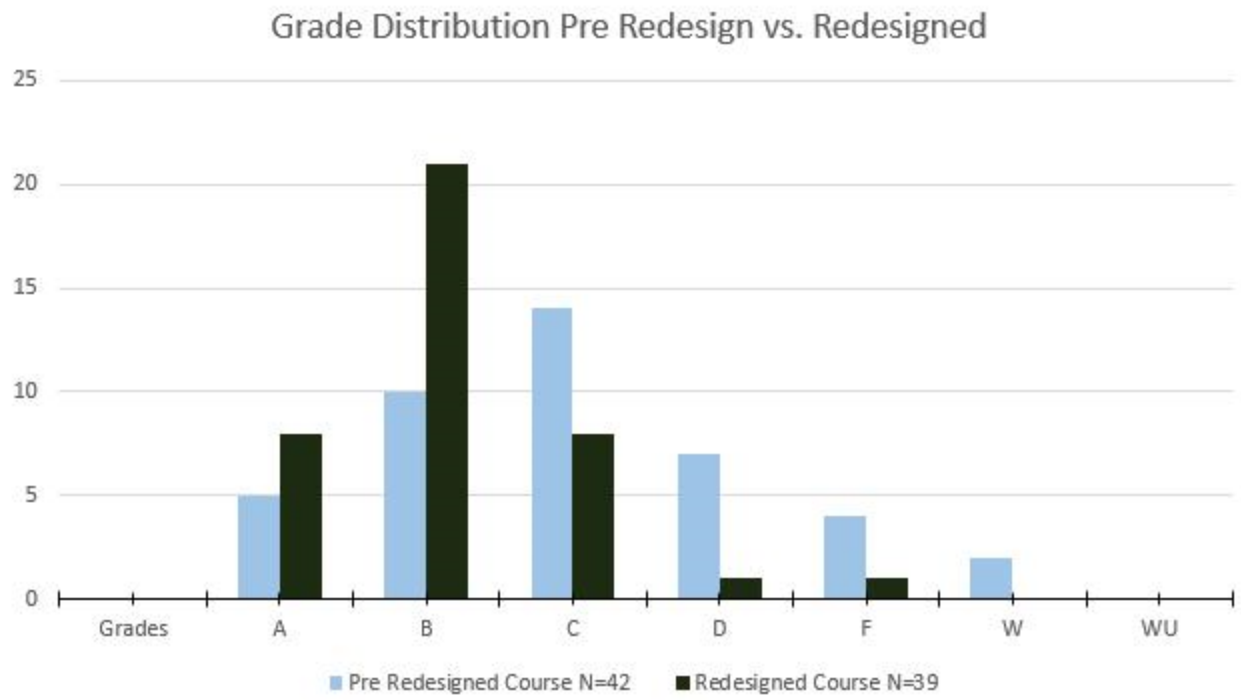

Figure 1: Pre-Redesign (non-SI) vs. redesign (SI) grade distributions.

	Pre Redesigned Course N=42	Redesigned Course N=39
A	5	8
B	10	21
C	14	8
D	7	1
F	4	1
W	2	0
WU	0	0
Total	42	39

Note: D and Lower are not passing

Table of Actual Count for Pre and Post Redesign

Table 1: Pre and post redesign of course grade counts.

Table 1 shows the raw count of grades for both the pre and post redesigns. As can be seen, the curve shifts positively in grade direction as well as higher count of non-DFW grades for the version utilizing the redesign. Specifically the DFW rate changes from 31% to less than 1%. A t-test performed on the two samples at a significance level of 0.05 results in a p-value of 0.003, indicating a statistically significant difference between the two distributions. Another indicator associated with student performance that was analyzed was the student evaluations. These evaluations ask students a number of questions regarding the course that are scored from high-to-low on a scale of 5.0-0. The average pre redesign version scored 3.2/5.0 while the post redesign scored 3.9/5.0. Some of the questions asked in the evaluations involved: the students increase in overall knowledge of the subject, instructor presentation understandability, instructor preparation, instructor feedback, and assignment contribution to learning. Hence, the increase in

evaluation scores is weakly attributed to SI and more so to the teaching method. However, both of these data sets harness a significant jump, indicating either the new lecture format, SI, or a combination of both contributed.

Conclusion and Future Work

Data comparing implementation of a redesigned tablet-based hand-written lecture format and SI into the same "Introduction to Technical Computing" course indicate a positive shift in the DFW rate as compared to without SI. There is a statistically significant difference between the two grade distributions. Students also seem to rate the redesigned course more positively in their evaluations.

Further data collection is ongoing to reinforce these results, in addition a course with the new technology and no SI is underway to try and isolate the effects of each component of the redesign. Implementation of SI was not addressed in the current student evaluations, so adding this to the evaluations could provide useful feedback. Additional data that could be of use would be to test the various implementations with a different course instructor, and look for similar performance differences. In conclusion, when some of principle problems associated with teaching and learning programming are analyzed, modified lecture with SI seems to offer some positive initial results.

References

[1] D. Sleeman, "The challenges of teaching computer programming," Communications of the ACM, Vol. 29, No. 9, 1986.

[2] S. Sentance and A. Csizmadia, "Computing in the curriculum: Challenges and strategies from a teacher's perspective," Educ. Inf. Technol., Vol. 22, pp.469-495, 2017.

[3] M. Ben-Ari, "Constructivism in computer science education," *Proceedings of the twenty-ninth SIGCSE technical symposium on computer science education*. Atlanta, Georgia, United States: ACM, 1998.

[4] M. Hurley, G. Jacobs, and M. Gilbert, "The basic SI model," New Directions for Teaching and Learning, Vol. 2006, Issue 106, pp. 11-22, 2006.

[5] A. Bandura, "Social Learning Theory." Prentice-Hall, Englewood Cliffs, NJ, 1977.

[6] L. Flower and J.R. Hayes, “A cognitive Process Theory of Writing,” College Composition and Communication, Vol. 32, pp. 365-387, 1981.

[7] C. Geertz, “Local Knowledge: Further Essays in Interpretive Anthropology.” Basic Books, New York, 1983.

[8] M. Apple, “Teachers and Text.” Routledge, New York, 1988.

[9] A. K. Ribera, A. BrckaLorenz, and T. Ribera, “Exploring the Fringe Benefits of Supplemental Instruction,” Association for Institutional Research Annual Forum, 2012.

[10] J. Malm, L. Bryngfors, and L. Morner, “Benefits of Guiding Supplemental Instruction Sessions for SI Leaders: a Case Study for Engineering Education at a Swedish University.” Journal of Peer Learning, Vol. 5, 2012