2018 ASEE ANNUAL CONFERENCE & EXPOSITION
JUNE 24–27, 2018
SALT PALACE CONVENTION CENTER
SALT LAKE CITY, UT

Paper ID #23173

IUSE Computational Creativity: Improving Learning, Achievement, and Retention in Computer Science for CS and non-CS Undergraduates

Markeya S. Peteranetz, University of Nebraska, Lincoln
Dr. Duane F. Shell, University of Nebraska, Lincoln

Duane Shell is Research Professor of Educational Psychology at the University of Nebraska-Lincoln. His primary research areas are learning, self-regulation, and motivational influences on behavior and cognition as these are manifest in education and public health settings. Dr. Shell specializes in multivariate, multidimensional analyses of complex relationships between motivation, classroom factors, self-regulation, and learning. He is primary author of the Unified Learning Model. In addition to his primary research, he has 32 years experience as an evaluator on federal, state, and foundation grands.

Prof. Leen-Kiat Soh, University of Nebraska, Lincoln

Dr. Leen-Kiat Soh is a Professor at the Computer Science and Engineering Department at the University of Nebraska. His research interests are in multiagent systems, computer-aided education, computer science education, and intelligent image analysis. He has applied his research to smart grids, computer-supported collaborative learning, survey informatics, geospatial intelligence, and intelligent systems, and He is a member of IEEE, ACM, and AAAI.

Dr. Elizabeth Ingraham, University of Nebraska, Lincoln

Elizabeth Ingraham is Associate Professor in the School of Art, Art History & Design at the University of Nebraska-Lincoln. A Fellow of the Center for Great Plains Studies, she teaches design and computational creativity at UNL and received the Sorensen Award for excellence in humanities teaching. A sculptor whose work gives form and voice to lived experience, she won the Thatcher Hoffman Smith Award for Creativity for her series of life-size sewn fabric ”skins” sculptures. Her recent solo exhibition at the International Quilt Study Center & Museum showcased the result of more than 9,000 miles of travel across Nebraska for her project, ”Mapping Nebraska”—a stitched, drawn and digitally imaged cartography of the state (physical and psychological) where she resides. Her research into computational creativity is part of her on-going interest in combining the digital (pixels and code) with the digital (the work of the hand).

Mr. Abraham Flanigan

IUSE Computational Creativity: Improving Learning, Achievement, and Retention in Computer Science for CS and non-CS Undergraduates

Introduction and Project Background

This project is funded through the NSF Improving Undergraduate STEM Education (IUSE) initiative and seeks to enhance undergraduate computer science (CS) education by teaching computational creativity in both CS and non-CS courses. The purpose of this paper is to present the methods used in this project, summarize previous findings, and report new results related to students' retention in CS courses. Computational creativity integrates computational thinking and creative thinking so that each can be used to enhance the other in improving student learning and performance in class [1]. Whereas computational thinking brings a structured and analytic approach to problem-solving situations, creative thinking introduces novelty and innovative, non-standard solutions.

While numerous components of computational thinking have been identified (e.g., [2]), the focal components within our computational creativity framework are abstraction, algorithmic thinking, evaluation, generalization, pattern recognition, and problem decomposition. The theory of creativity that undergirds computational creativity is Epstein's creative competencies [3]. Epstein's four core competencies are broadening, capturing, challenging, and surrounding. Descriptions of the components of computational thinking and the creative competencies are given in Table 1.

Table 1. Components of Computational Thinking and the Creative Competencies

Component	Description
Computational Thinking	
Abstraction	Reducing complexity by identifying general rules and principles that involve only essential elements
Algorithmic thinking	Generating procedural rules that can simplify a process
Evaluation	Testing a solution's effectiveness
Generalization	Applying existing processes and solutions to new problems
Pattern recognition	Identifying common characteristics and recurring elements
Problem decomposition	Breaking down a problem into smaller chunks that can be addressed separately
Creative Thinking	
Broadening	Building one's knowledge base beyond one's discipline
Capturing	Preserving ideas and solutions
Challenging	Questioning conventions and moving beyond established thinking and behavior patterns
Surrounding	Seeking out and immersing oneself with new social and environmental stimuli

In our project, computational creativity is introduced to undergraduate students through Computational Creativity Exercises (CCEs). CCEs are collaborative problem-solving tasks that require teams of students to work together to apply computational and creative thinking to novel situations. Although the CCEs are heavily steeped in computational principles and were first

introduced in CS courses, the exercises do not use any programming code and involve tasks not inherently connected to CS. For example, in the Storytelling exercise, students write small "chapters" of a story individually and then after reading each team members' chapter, they "de-bug" the whole story by revising their own chapters to make the whole story consistent.

To date, our research team has developed more than a dozen CCEs (see Table 2)—with numerous variants—that have been implemented in introductory, intermediate, and advanced CS courses as well as non-CS courses, including a stand-alone Computational Creativity course. The Computational Creativity course is offered through our University's School of Art, Art History, and Design, and in the course students are taught how to combine creative thinking and computational thinking in problem solving and produce creative artifacts. Students in all these courses complete the CCEs through an on-line platform that allows for collaborative work and communication throughout the exercises. Variants of the CCEs can be found at Google's Exploring Computational Thinking repository, EngageCSEdu site, and the Project Ensemble's Portal. We encourage interested parties to contact us for more information on the CCEs.

Table 2. Computational Creativity Exercise Descriptions

Name	**Brief Description**
Everyday Object	Identify an "everyday" object (such as nail clipper, a paper clip, Scotch tape) and describe the object in terms of its inputs, outputs and functionalities.
Cipher	Devise a three-step encoding scheme to transfer the alphabet letters into digits and encode questions for other teams to compete to decode.
Story Telling	Develop a chapter (100-200 words) individually and independently in week 1 and work as a team in week 2 to resolve all conflicts or inconsistencies.
Exploring	Explore sensory stimuli at a particular site (sounds, sights, smell, etc.) and document observations.
Simile	Poses "simile" descriptions and participate in team-to-team Q&As to solicit guesses and descriptions relevant to a particular object.
Machine Testing	Devise ways to test a black-box mysterious machine without causing harm to humans while attempting to reveal the functionalities of the machine.
Calendar	Build a calendar for a planet with two suns, four different cultural groups with different resource constraints and industrial needs.
Path Finding I	Create a step-by-step instruction on drawing lines to create a quilt pattern on a n x n grid and identify similar structures in other teams' quilt patterns.
Path Finding II	Use rotation, reflection, and loop to generate a more complex quilt pattern based on simpler base pattern.
Marble Maze I	Each team member creates a sub-structure allowing a marble to travel at least for n seconds in week 1 and the team puts all sub-structures together to make a super-structure in week 2.
Marble Maze II	Teams are broken up and now must adapt their own sub-structure to work with other sub-structures in their new teams.
Marble Maze III	All teams bring together their super-structures and build a mega-structure.
Big Five Profiles	Revises a text snippet such that at least one the text snippet's Big Five profile changes significantly
Dividing Alphabet	Finds a rule to divide up the alphabet letters based on some sample data points on how some initial letters are divided.

Method Summary

Over several semesters, we have collected data in classes where the CCEs were implemented as well as control classes that did not include the CCEs. When CCEs are implemented, they are used to supplement instruction and do not replace other homework assignments or learning activities. Initially, the research team facilitated the CCEs and graded students' work, but in more recent semesters, course instructors have had greater responsibility in the implementation. Students have reacted positively to the full inclusion of the CCEs in the courses and increased alignment in timing between the CCEs and instruction on relevant topics [4]. In all classes, students complete surveys at the beginning, middle, and end of the semester that assess their motivation, strategic regulation, affective reactions to the course, and, in the implementation classes, reactions to each of the CCEs. The end-of-semester survey includes a 13-item test of core CS concepts and computational thinking that is used to compare students' CS knowledge across classes [5]. Students also have the option of giving us permission to obtain their course grades and grade point average (GPA) at the end of the semester and course enrollment for three semesters after the semester during which they participated in the study.

Summary of Previous Studies

Previously reported findings [4],[6]–[11] provide evidence that CCEs can improve undergraduate students' learning and performance in CS classes at all course levels. In [6]–[7] we reported a "dosage effect" that indicated student learning in introductory CS increased linearly with each additional CCE completed. Miller et al. [8] replicated this finding with a sample that comprised students in introductory, intermediate, and advanced CS courses. And, findings reported in [9] indicate the effect of the CCEs is independent of students' general academic achievement, motivation, engagement, and strategic self-regulation.

Quasi-experimental investigations of engineering students in introductory CS have yielded similar results [4],[9]–[10] . In two studies [9]–[10], we compared CS knowledge test scores of engineering students in introductory CS taught with the CCEs to the scores of engineering students in introductory CS taught without the CCEs during a different semester. Students in the classes taught with the CCEs scored higher on the CS knowledge test than students in the control sections, indicating the CCEs contributed to learning in the introductory CS course.

This finding was replicated with two more rigorous quasi-experimental studies that used propensity score matching to equate the implementation and control groups on motivation variables [4],[11]. In one study [11], CCEs were implemented in lower- and upper-division CS courses, and students in those courses were matched with students in the same courses when the CCEs were not implemented. Students in the implementation group had higher grades and scores on the CS knowledge test, and these effects were similar in lower- and upper-division courses. In the other study [4], CCEs were implemented in one of two sections of the same introductory CS class for engineers taught during a single semester. The instructors of the two sections coordinated and synchronized their lecture topics, shared their lecture notes throughout the semester, and met weekly—with their shared teaching assistants—to discuss issues related to student learning and course activities. Additionally, the two sections shared laboratory sections

and used the same graded assignments and tests. Results of this study again showed that students in classes with CCEs score higher on the CS knowledge test than students in non-CCE classes, further supporting the hypothesis that CCEs contribute to learning core CS concepts.

Recent Findings

The most recent extension of our project is the investigation of the impact of CCEs on students' retention in CS courses. We hypothesize that exposure to CCEs will increase the likelihood that students will continue to take CS classes. In the present quasi-experimental study, CCEs were implemented in 100- and 200-level CS courses at a single university during the fall of 2015. Students in these courses (implementation group; $N = 670$) consented to having their course enrollment data collected for the following three semesters (retention semesters). Students in 100- and 200-level CS courses during the fall of 2014 and spring of 2015 semester also consented to having their ongoing course enrollment data collected and were used as a comparison (control) group.

The impact of the CCEs on retention was tested through chi square (χ^2) analysis for each of the three retention semesters. Results indicate students in the implementation semester courses were more likely than students in the control semester courses to continue taking CS in each of the three retention semesters (see Table 3), and the effect was even greater when comparing students who did no CCEs (in either the implementation or control groups) and those who completed at least two CCEs (see Table 4). Thus, these results suggest that in addition to the impact CCEs have on achievement, CCEs also increase the likelihood that CS students will continue to take CS courses.

Table 3. Impact of CCEs on Retention Based on Enrollment in Implementation Courses

	χ^2	*p*	Cramer's V
Semester 1	13.88	<.001	.144
Semester 2	21.87	<.001	.193
Semester 3	12.97	<.001	.153

Table 4. Impact of CCEs on Retention Based on CCE Completion (at least two exercises)

	χ^2	*p*	Cramer's V
Semester 1	18.89	<.001	.175
Semester 2	34.84	<.001	.254
Semester 3	25.87	<.001	.227

Conclusion

Computational Creativity Exercises have been shown to increase undergraduate students' achievement in CS courses, and new findings indicate they might also increase the likelihood that students will continue taking CS courses. The effect on retention is notable. Considering the ongoing concerns about retaining computer science students, the value of learning activities that boost achievement *and* increase retention is considerable. The finding that the retention boost is significant and lasts at least three semesters sets CCEs apart from other researched instructional practices because, to our knowledge, no other instructional practices have been shown to increase retention in CS.

References

[1] L. K. Soh, D. F. Shell, E. Ingraham, S. Ramsay, and B. Moore, "Learning through computational creativity," *Commun. ACM*, vol. 58, no. 8, pp. 33-35, Aug. 2015.

[2] J. Wing, "Computational thinking," *Commun. of the ACM*, vol. 49, pp. 33-35, Mar. 2006.

[3] R. Epstein, S. Schmidt, and R. Warfel, "Measuring and training creativity competencies: Validation of a new test," *Creativity Res. J.*, vol. 20, pp. 7-12, Feb. 2008.

[4] M. S. Peteranetz, A. E. Flanigan, D. F. Shell, and L.-K. Soh, "Helping engineering students learn in introductory computer science (CS1) using computational creativity exercises (CCEs)", *IEEE Transactions on Education*, advance online pulication, pp.1-9, 2018. doi: 10.1109/TE.2018.2804350

[5] M. S. Peteranetz, A. E. Flanigan, D. F. Shell, and L.-K. Soh, "Career aspirations, perceived instrumentality, and achievement in undergraduate computer science courses," *Contemporary Educational Psychol.*, vol. 53, pp. 27-44, 2018.

[6] L. D. Miller *et al*., "Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses," in *Proc. IEEE Frontiers Edu. Conf.*, Oklahoma City, OK, USA, 2013, pp. 1426-1432.

[7] D. F. Shell, M. Patterson-Hazley, L. K. Soh, E. Ingraham, and S. Ramsay, "Impact of creative competency exercises in college computer science courses on students' creativity and learning," presented at the Annu. Meeting of the Amer. Educational Res. Assoc., Philadelphia, PA, USA, Apr. 3-7, 2014.

[8] L. D. Miller, L. K. Soh, and D. F. Shell, "Integrating computational and creative thinking to improve learning and performance in CS1," in *Proc. 45th ACM Technical Symp. Comput. Sci. Edu.,* Atlanta, GA, USA, 2014, pp. 475-480.

[9] M. S. Peteranetz, A. E. Flanigan, D. F. Shell, and L.-K. Soh, "Computational creativity: An avenue for promoting learning in computer science, *IEEE Transactions on Education*, vo. 60, no.4, pp. 305-313, 2017.

[10] D. F. Shell *et al*., "Improving learning of computational thinking using computational creativity exercises in college CS1 computer science course for engineers," in *Proc. IEEE Frontiers Edu. Conf.*, Madrid, Spain, 2014, pp. 3029-3035.

[11] M. S. Peteranetz, S. Wang, D. F. Shell, A. E. Flanigan, and L.-K. Soh, "Examining the impact of computational creativity exercises on college computer science students' learning, achievement, self-efficacy, and creativity," in *Proc. 49th ACM Technical Symposium on Computer Science Education*, Baltimore, MD, USA, 2018, pp. 155-160.