EngageCSEdu Teaching Paper

Implementing UNL's Computational Creativity Exercises

Leen-Kiat Soh, Elizabeth Ingraham, Duane Shell, Markeya Peteranetz, Lee Dee Miller and Abraham Flanigan
University of Nebraska, Lincoln, NE

Abstract

This teaching paper provides an overview of the Computational Creativity Exercises (CCEs) that can be used in introductory computing courses classrooms to improve student learning and performance in computational thinking. The exercises are group-based, unplugged activities. This encourages students to practice thinking and communicating their ideas, without having to worry about coding and syntax, and be constrained to the programming environment. The goal of this paper is to provide other instructors with guidance on implementing these exercises by addressing logistical issues and providing tips for adopting and adapting for individual classroom needs. As part of this overview, we summarize research findings from our multi-year studies on the impact of these exercises.

1. INTRODUCTION

The premise behind integrating **computational thinking** and **creative thinking** is that they complement each other in improving student learning and performance in class [1]. Whereas computational thinking brings a structured, convergent, and analytic approach to problem-solving situations, creative thinking introduces novelty and innovative, divergent, non-standard solutions. While numerous components of computational thinking have been identified (e.g., [2]), the skills that we focus on in these exercises are abstraction, algorithmic thinking, evaluation, generalization, pattern recognition, and problem decomposition. The theory of creativity underlying these Computational Creativity Exercises (CCEs) is Epstein's core creative competencies [3]: broadening, capturing, challenging, and surrounding. In addition, the CCEs also include five **collaborative activities** such as collaboration, communication, coordination, persistence, and play. Descriptions of the components of computational thinking, the creative competencies, and collaborative activities are given in Table 1.

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. Variants of the CCEs can be found at the EngageCSEdu site, Google's Exploring Computational Thinking repository, and the Project Ensemble's Portal. We encourage interested parties to contact us for more information on the CCEs.

Table 1. Components of Computational Thinking, the Creative Competencies, and Collaborative Activities

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
Collaborative Activities	
Collaboration	Sharing ideas and participating in group dialogue, and being open-minded and flexible to work as a team to implement the most effective solution
Communication	Sharing ideas and viewpoints and practicing empathy and reframing in order to understand and respond to others' approaches
Coordination	Scheduling and planning group work and adhering to deadlines
Persistence	Pushing beyond the easy solution or the conventional response, and using critical thinking to revise solutions to make them more effective and to learn from failure
Play	Imagining, testing, and tinkering without imposing unnecessary constraints, and experimenting and investigating connections and inspirations from diverse sources

Table 2. Computational Creativity Exercise Descriptions

Name	Brief Description
Everyday Object[1]	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[1]	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	Pose "simile" descriptions ("It is like the sky") 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 loops to generate a more complex quilt pattern based on a simpler base pattern.

[1] This CCE can be found in the EngageCSEdu collection.

Marble Maze I	Each team member creates a sub-structure allowing a marble to travel for at least n seconds in week 1 and the team puts all sub-structures together to make a super-structure in week 2.
Marble Maze II	Team members are shuffled 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	Revise a text snippet such that at least one the text snippet's Big Five profile changes significantly
Dividing Alphabet	Find a rule to divide up the alphabet letters based on some sample data points on how some initial letters are divided.

2. COMPUTATIONAL CREATIVITY EXERCISES

Each CCE has four common components: **Objectives**, **Tasks**, **CS Lightbulb** (e.g., explanations connecting activities to CS concepts, ideas, and practices) and **Reflection and Analysis Questions**. As depicted in Table 2, CCE problems cover a wide range of problems with varying degrees of obvious connection to CS, which allows for CCEs at differing levels of abstraction. Students answer analysis and reflection questions designed to further promote both computational thinking and the creative application of computational skills. The CCEs are designed so that the students have hands-on and group tasks first, in Part 1, and then reflect on their Part 1 activities in Part 2 by answering questions. Both parts are graded. Depending on how one administers the CCEs, each CCE can be done in class in two 1-hour periods, or outside class, say, for two weeks.

Our CCEs are anchored in instructional design principles shown to impact deeper learning, transfer, and development of interpersonal skills. They are designed to provide instruction on concepts related to computational thinking and computer science (CS) combining hands-on problem-based learning (PBL) with written analysis and reflection. They facilitate transfer by utilizing computational thinking and CS content more abstractly and without using programming code to address problems seemingly unrelated to CS. By requiring groups of students to work collaboratively, the exercises draw on the diverse backgrounds of the students. The CCEs foster development of creative competencies by engaging multiple senses, requiring integrative, imaginative thought, presenting challenging problems and developing interpersonal skills using individual and collaborative group efforts.

3. SUMMARY OF PREVIOUS STUDIES

A series of studies demonstrates the effectiveness of CCEs in promoting learning and performance in undergraduate CS classes at all course levels. In [5, 6] we reported a "dosage effect" that indicated student learning in introductory CS increased linearly with each additional CCE completed. This finding was replicated [7] with students in introductory, intermediate, and advanced CS courses. And, this effect appears to be independent of general academic achievement, motivation, engagement, and strategic self-regulation [8].

Quasi-experimental studies of engineering students in introductory CS have similarly shown positive effects of CCEs [4, 8, 9]. In two studies [8, 9], we compared engineering students in introductory CS taught with the CCEs to engineering students in the same course taught without the CCEs during a different semester. Students in the classes taught with the CCEs scored higher on a CS knowledge test than students in the control sections, indicating the CCEs contributed to learning in the introductory CS course.

The positive effect on learning was replicated with two more rigorous quasi-experimental studies that used propensity score matching (PSM) to equate the implementation and control groups on motivation variables [4, 10]. In one study [10], 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 used. Students in the implementation group had higher grades and scores on a CS knowledge test, and the effects were consistent in both lower- and upper-division courses. In the other PSM study [4], CCEs were implemented in one of two sections of the same introductory CS class for engineers taught during the same semester. The two instructors 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. The two sections also shared laboratory sections and used the same graded assignments and tests. Results again showed that students in classes with CCEs scored higher on a CS knowledge test than students in non-CCE classes, further indicating that CCEs contribute to learning core CS concepts.

4. LOGISTICS

When CCEs were fully integrated into the class, students enjoyed them or appreciated the rationales behind the CCEs. Here are some recommendations on the logistics of administering or deploying these CCEs:

- Discuss each exercise in class (5-10 minutes).
- Explicitly map activities in the exercise to topics being learned in class.
- Relate both computational and creative thinking objectives to real-world problems.
- To encourage student participation, the exercises should be graded.
- Count 3-5% towards the final course grade depending on the number of exercises you assign. If the CCEs are fully integrated into the course, consider assigning points equivalent to a regular lab or homework assignment.
- Adapt reflection and analysis questions to help students gain insights and "meta-knowledge."
- Adapt lightbulbs to meet your needs. These are important for connecting back to explicit course material and for addressing the concerns some students might (wrongly) have about these exercises being non-technical or non-CS.

In-Class vs. Out-of-Class. As noted earlier, it is possible to administer the exercises as in-class activities entirely. In this scenario, each student group works together to complete Part 1 in the first lecture or period, and then comes together again to complete Part 2 in the second lecture or period. At the end of each lecture or period, each group submits their required documents such as an essay or a filled-out form. It is also possible to administer the exercises as out-of-class activities with instructors providing some in-class introduction and connection to topics covered in class (see example implementation below). In this scenario, student groups can be directed to participate or collaborate via a Learning Management System (LMS) platform that offers forum discussions and collaborative wiki writing features. Students in each group can then interact with the LMS platform to discuss, coordinate, create, and revise their written products, and so forth. The Part 1 and Part 2 deliverables and deadlines can then be adjusted accordingly. For example, if students are from different backgrounds and disciplines and that group meetings might be difficult to schedule, then relying on the LMS for collaboration might take additional time. In general, if the size of a class is small and seating arrangement is flexible, then an instructor might have sufficient time to conduct the assignment entirely in-class.

Example Out-of-Class Implementation. Before assigning the exercise, the instructor prepares by reviewing the exercise, paying particular attention to the lightbulbs and deadlines. The instructor can

revise lightbulbs to connect more closely to course content and should create project timelines that promote application of concepts recently covered in class. Next, the instructor forms the groups. A group of 3-5 students should be sufficient; if the group size is too small, then the benefits of collaborative activities are diminished, but if the group size is too big, then there are potentially coordination issues (e.g., it will be more difficult to find common times to meet). The groups can be formed randomly or in a stratified manner such that there is diversity in each group. For example, in a class of students from different majors, one could design each group to have students from at least three different majors.

When the exercise is assigned, the instructor goes over the exercise and its activities in class and uses the lightbulbs to connect to topics covered in class. This could take 5-15 minutes. The instructor then sends out a reminder e-mail before Part 1's deadline, for example, to encourage discussions and completions. After Part 1's deadline, the instructor can preview some of the submissions and then discuss again in class for 5-10 minutes. Prior to Part 2's deadline, the instructor sends out another reminder e-mail. Finally, after the exercise is completed and graded, the instructor then reports on some of the discussions and results. For example, the instructor could share with the class good and bad examples of abstraction and generalization from the Everyday Objects essay.

Group Management. Despite the challenges of collaborating online and the potential for friction between group members, group work has many advantages. For example, considering different points of view and resolving conflicts is part of the exercise objectives. Learning how to have "creative abrasion" while avoiding interpersonal conflict is a key workplace skill. Groups can tackle bigger tasks than individuals, and randomly assigned groups offer diversity and can allow every student to speak.

Additional recommendations on the logistics of facilitating group work include:

- Consider revising groups if some students' lack of participation becomes a problem.
- Consider giving groups the opportunity to meet in class; even 10 minutes can go a long way to encourage community and to get questions to surface so that the procedure can be clarified.
- Provide context to students (even 10 minutes) in class and point out direct connections to course content both before and after the exercise to increase students' positive perception of the activity and their participation.
- Consider designating a group leader for each group based on student performance in class, as high-performing students would tend to continue to want to perform well and be responsible for managing his or her group. Further, this also could be an opportunity to increase the number of female group leaders.

Grading. In terms of grading, we have adopted the following practices:

- If the exercises are more for extra credit, then grading is Yoda-esq ("Do or do not. There is no try"): as long as students show work, we give them credit with clearly defined grading rubrics that require evidence of participation in the exercise activity (e.g., labeled contribution or version history).
- If the exercises are given a number of points closer to that given to a lab or homework assignment, then grading is graded in a more typical manner for an assignment.
- Analysis and Reflection questions require 3-5 relevant sentences and a response to another student's reflection/analysis (also 3-5 relevant sentences). Students post their own responses before seeing others' responses. In this manner, students help keep their group members accountable.

Potential Pitfalls. In addition to pitfalls associated with group assignments, there are two other sets of pitfalls worth mentioning here: (1) exercise-related, and (2) platform-related. Exercise-related pitfalls

include the following. First, as alluded to above, if an exercise is not connected to some topic(s) in class, students will be disengaged and perceive the exercise as something not part of the course and view it less seriously. Further, there are exercises (such as Cipher, Simile, and Pathfinding) that require groups to respond to other groups' essays or questions. If deploying these exercises as first exercises in the semester, inter-group interactions might get chaotic as a student would have to learn to work with his or her own group members *and* respond to outside-group inquiries in the same assignment. Furthermore, should artifacts from previous semesters be made available to students? These artifacts could serve as inspiring examples, but at the same time could also lead to convergent thinking and reduce creativity. In our deployment, we typically share one or two exemplary artifacts. Platform-related pitfalls include the following. First, students might not be familiar with the platform and an on-boarding exercise might need to be administered to get them started. We typically have one such exercise to get each group of students to communicate among them and post a message or two on the collaborative LMS platform. Second, not all students will use the LMS platform to collaborate, as they might meet in person or via another online social platform. Thus, the lack of observable activities on the LMS platform does not mean that a group has not been active.

5. ADAPTATION AND DEVELOPMENT

We encourage ***adaptations*** of these CCEs to suit individual classroom needs. For example, tasks can be reduced in scale or scope, and the questions can be reduced or replaced. Lightbulbs should be reviewed and adapted to best connect the activities to the topics covered in your class. However, core components—the three sets of objectives, the tasks, the lightbulbs, and the reflection and analysis questions— should be retained, as they are key.

We encourage others to develop new exercises. The CCEs that we have created thus far have started as an idea, or a game, or a puzzle, or an observation of a daily activity. For example, a game idea would be for a person A to describe a picture to a person B, and for the person B to relay that description to a person C and have C draw a picture based on the description, then compare the two pictures, identify the differences, and analyze the missing parts in the description. After identifying the tasks, we would write the lightbulbs to connect specific activities to the topics covered in class. Subsequently, we would develop questions inspired by the activities and lightbulbs, as well as by students' anticipated misconceptions or approaches. For example, if there is a lightbulb that ties the activities to a concept X, then a reflection question can be in the following form: "During your process of accomplishing the activities, did you gain insights into the concept X and, if yes, how?" Also, if there is a common misconception Y, then "Did your group perform Y? If yes, do you think it was a good or correct idea? If no, why not?" Meanwhile, as we develop these three components, we start drafting the three sets of objectives and revise them accordingly to reflect the activities, lightbulbs, and questions.

Note that these exercises are designed to be unplugged with no programming activities. As alluded to in the Introduction, this frees the students up from programming. Furthermore, because these exercises seemingly have nothing to do with programming and computer science, they also provide a learning opportunity for instructors and students to see problem solving in a more abstract and generalizable context.

6. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation grants no. DUE-1122956 (NSF TUES) and DUE-1431874 (NSF IUSE). Additional support was provided by a UNL Phase II Pathways to Interdisciplinary Research Centers grant.

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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

For more information about the University of Nebraska's Computational Creativity Exercises project and related research, see http://cse.unl.edu/agents/ic2think/.