DOI:10.1145/2699391 Leen-Kiat Soh, Duane F. Shell, Elizabeth Ingraham, Stephen Ramsay, and Brian Moore

Viewpoint Learning Through Computational Creativity

Improving learning and achievement in introductory computer science by incorporating creative thinking into the curriculum.

THE NATIONAL SCIENCE Foundation's "Rebuilding the Mosaic" report[a] notes that addressing emerging issues in all fields will require utilization and management of large-scale databases, creativity in devising data-centric solutions to problems, and application of computational and computer tools through interdisciplinary efforts. In response to these needs, introductory computer science (CS) courses are becoming more than just a course for CS majors. They are becoming multipurpose: designed not just to prepare future CS scientists and practitioners, but also to inspire, motivate, and recruit new students to CS; provide computational thinking tools and CS skills to students in other disciplines; and even train future CS K–12 teachers. This multifaceted nature of introductory CS calls for new ways of envisioning the CS curriculum. Along with computational thinking, creativity has been singled out as critical to addressing important societal problems and central to 21st century skills (for example, the 2012 National Research Council report). Driven by these observations, we see an opportunity to revise introductory CS courses by explicitly integrating creative thinking into the curriculum.

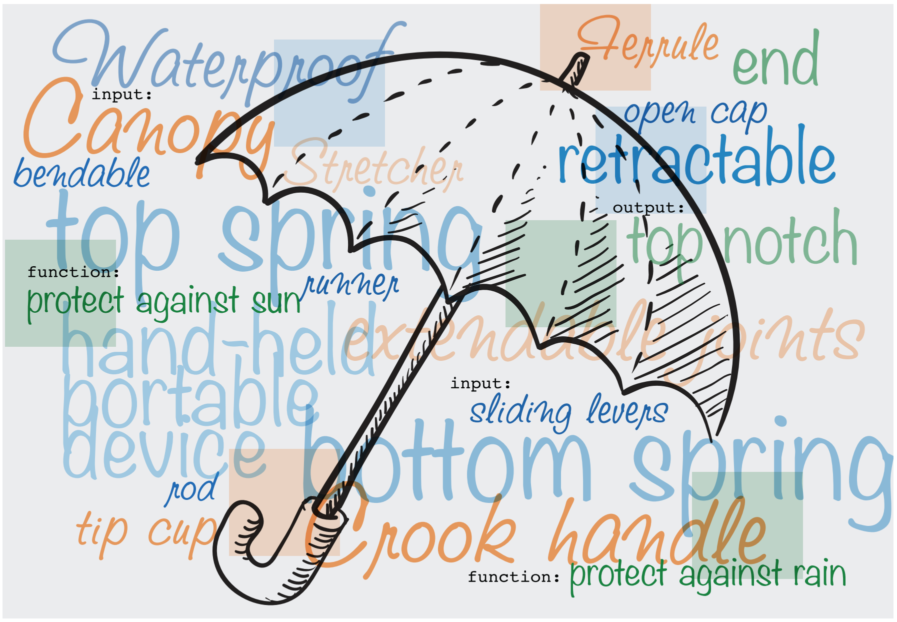
Creative Thinking

Creative thinking is not an innate talent or the province of just a few individuals, and it is not confined to the arts. Rather, it is a process integral to human intelligence that can be practiced, encouraged, and developed within any context.[1,2,5,7–9] Epstein's Generativity Theory[2] breaks creative thinking down to four core competencies:

- *Broadening.* The more diverse one's knowledge and skills, the more varied and interesting the possible novel patterns and combinations that might emerge. To be creative one must broaden one's knowledge by acquiring information and skills outside one's current domains of study and expertise.
- *Challenging.* Novelty emerges from situations where existing strategies and behaviors are ineffective. The more difficult the challenge, the more likely a creative, novel solution will emerge.
- *Surrounding.* Exposure to multiple, ambiguous situations and stimuli create environments where novel strategies and behaviors may emerge—for example, looking at things in new ways, interacting with new people, and considering multiple sensory representations.
- *Capturing.* Novelty is occurring all the time, but most of it passes without recognition. Creativity requires attention to and recording of novelty as it occurs.

These core competencies have a solid anchoring in contemporary cognitive and neuroscience research.[6] Just as Wing[10,11] makes a convincing case for the universality of computational thinking, we argue that Epstein's core creative thinking competencies are also a universally applicable skill set that provides a foundation not only for applying one's existing knowledge and skills in creative ways, but also for engaging in lifelong learning to broaden one's capabilities for work in interdisci-

a See http://www.nsf.gov/pubs/2011/nsf11086/nsf11086.pdf.



plinary environments on increasingly complex problems.

Computational Creativity Exercises

In our framework, both computational thinking and creative thinking are viewed as cognitive tools that when blended form *computational creativity*. This blending is not conceived as a dichotomy, but rather as symbiotic abilities and approaches. Computational thinking and CS skills expand the knowledge and tools that one has available, thereby *broadening* the scope of problem solutions. *Challenging* problems force computational tools to be used in unanticipated and unusual ways, leading to new computational approaches to both old and new problems. *Surrounding* oneself with new environments and collaborators creates novel ways of looking at problems and attention to different stimuli and perspectives that may be relevant to approaching a problem computationally. Finally, *capturing* ideas for data representations and algorithms can lead to new data structures and solution procedures. By merging computational and creative thinking, students can leverage their creative thinking skills to "unlock" their understanding of computational thinking.[6]

We have created a suite of Computational Creativity Exercises (CCEs) designed to increase students' computational creativity. Each CCE has four common components: Objectives, Tasks, CS Lightbulbs—explanations connecting activities to CS concepts, ideas, and practices—and Learning Objects that relate the exercise tasks directly to CS topics. The principles underlying the design of our Computational Creativity Exercises are balancing of attributes between computational and creative thinking and mapping between computational and creative concepts and skills as they manifest in different disciplines. Each CCE requires approximately one to two hours per student, but students are given two weeks to work on the exercises because of the collaboration required. The CCEs are designed so that the students have hands-on and group tasks first, in Week 1, and then reflect on their Week 1 activities in Week 2 by answering analysis and reflection questions. Both Week 1 and Week 2 are graded.

We see an opportunity to revise introductory CS courses by explicitly integrating creative thinking into the curriculum.

As an example, in the *Everyday Object CCE*, students are asked to act like the inventor of an ordinary object that we might frequently use. The challenge is to imagine this object does not exist and to describe in written language: the mechanical function of the selected object; what need is fulfilled by this object; and its physical attributes and characteristics. The description must be specific enough so that someone who had never seen the object could recognize it and understand how it works and understand what benefits it provides. (Note: Students were given a list of objects—zipper, mechanical pencil, binder clip, Ziploc bag, scissors, tape measure, stapler, nail clippers, umbrella, flashlight, can opener, clothespin, sticky notes, toilet paper holder, revolving door—from which to choose.) Students are then asked to consider and write their responses to the following questions. ***Analysis:*** (1) Consider your object as a computer program. Draw a diagram that shows all its functions as boxes (name them), and for each function, its inputs and outputs. Are there shared inputs and outputs among the functions? (2) Consider the list of physical attributes and characteristics. Organize these such that each is declared as a variable with its proper type. Can some of these attributes/characteristics be arranged into a hierarchy of related attributes/characteristics? ***Reflection:*** (1) Consider your response to Analysis 1, are there functions that can be combined so that the object can be represented with a more concise program? Are there new functions that should be introduced to better describe your object such that the functions are more modular? (2) Have you heard of abstraction? How does abstraction in computer science relate to the process of identifying the functions and characteristics as you have done in this exercise?

Our CCEs are anchored in instructional design principles shown to impact deep learning, transfer, and development of interpersonal skills. They are designed to provide instruction on CS concepts by combining hands-on problem-based learning with written analysis and reflection. They facilitate transfer by using computational thinking and CS content more abstractly and without using programming code to address problems seemingly unrelated to CS. 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. The CCEs engage the cognitive/neural learning processes of *attention*, *repetition*, and *connection* identified in the Unified Learning Model[6] on synthesis of research in cognitive neuroscience, cognitive science, and psychology. They enhance learning and retention of course material by focusing attention on computational thinking principles, provide additional repetition of computational thinking and computing concepts from the class, and provide connections of the material to more diverse contexts and applications at a higher level of abstraction.

These core competencies have a solid anchoring in contemporary cognitive and neuroscience research.

Some Evidence

Four CCEs were deployed in four different introductory computer science courses during the Fall 2012 semester at the University of Nebraska, Lincoln. Each course was tailored to a different target group (CS majors, engineering majors, combined CS/physical sciences majors, and humanities majors). Findings from 150 students showed that with cumulative GPA controlled, the number of CCEs completed was significantly associated with course grade ($F(3, 109) = 4.32$, $p = .006$, partial Eta^2 = .106). There was a significant linear trend ($p = .0001$) from 2 to 4 exercises completed. The number of CCEs completed also was significantly associated with a computational thinking knowledge test score ($F(3, 98) = 4.76$, $p = .004$, partial Eta^2 = .127), again with a significant linear trend ($p < .0001$) from 0–1 to 4 exercises completed with no differences for CS and non-CS majors.[3,4] These findings indicated a "dosage" effect with course grades and test scores increasing with each additional CCE completed. The increases were *not* trivial. Students increased by almost a letter grade and almost one point on the knowledge test per CCE completed.

In a second evaluation,[7] we employed a quasi-experimental design comparing CCE implementation in the introductory computer science course tailored for engineers during Spring 2013 ($N = 90$, 96% completing three or four exercises) to a control semester ($N = 65$) of no implementation in Fall 2013. Using Analysis of Covariance (ANCOVA), we found that students in the implementation semester had significantly higher computational thinking knowledge test scores than students in the control semester ($M = 7.47$ to $M = 5.94$) when controlling for students' course grades, strategic self-regulation, engagement, motivation, and classroom perceptions. ($F(1, 106) = 12.78$, $p <.01$, partial Eta^2 = .108). CCE implementation students also reported significantly higher self-efficacy for applying their computer science knowledge and skills in engineering than non-implementation students ($M = 70.64$ to $M = 61.47$; $F(1, 153) = 12.22$, $p <.01$, partial Eta^2 = .074).

Overall, in relation to traditional findings for classroom educational interventions, these are strong effects that demonstrate meaningful "real-world" impact. The exercises appear to positively affect the learning of core course content and achievement for both CS majors and non-majors. The findings support our contention that *the Computational Creativity Exercises can bring CS computational concepts to CS and non-CS disciplines alike and improve students' understanding of computational thinking*.

Call to Action

Encouraged by our evaluation findings, we are currently working on adapting the CCEs to secondary schools, designing a course based on a suite of CCEs, as well as continuing with the development of more CCEs. Furthermore, the broader impacts of incorporating computational creativity into introductory CS courses are wide-ranging including reaching underrepresented groups in CS, outreach exposing young learners to computational creativity, improving learning of CS, and preparing students to be creative problem solvers in increasingly interdisciplinary domains. We thus call to action CS (and other STEM) educators to investigate and adopt computational creativity in their courses or curricula. **C**

References

1. Epstein, R. *Cognition, Creativity, and Behavior: Selected Essays.* Praeger, 1996.
2. Epstein, R. Generativity theory and creativity. *Theories of Creativity.* Hampton Press, 2005.
3. Miller, L.D. et al. Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses. *FIE* 43, (2013), 1426–1432.
4. Miller, L.D. et al. Integrating computational and creative thinking to improve learning and performance in CS1. *SIGCSE'2014* (2014), 475-480.
5. Robinson, K. *Out of Our Minds: Learning to be Creative.* Capstone, 2001.
6. Shell, D.F., Brooks, D.W., Trainin, G., Wilson, K., Kauffman, D.F., and Herr, L. *The Unified Learning Model: How Motivational, Cognitive, and Neurobiological Sciences Inform Best Teaching Practices.* Springer, 2010.
7. Shell, D.F. et al. Improving learning of computational thinking using computational creativity exercises in a college CS1 computer science course for engineers. *FIE* 44, to appear.
8. Shell, D.F. and Soh, L.-K. Profiles of motivated self-regulation in college computer science courses: Differences in major versus required non-major courses. *J. Sci. Edu. Tech. Technology* (2013).
9. Tharp, T. *The Creative Habit: Learn it and Use it for Life.* Simon & Schuster, 2005.
10. Wing, J. Computational thinking. *Commun. ACM 49*, 3 (Mar. 2006), 33–35.
11. Wing, J. Computational thinking: What and why. *Link Magazine* (2010).

Leen-Kiat Soh (lksoh@cse.unl.edu) is an associate professor in the Computer Science and Engineering Department at the University of Nebraska.

Duane F. Shell (dshell2@unl.edu) is a research professor at the University of Nebraska.

Elizabeth Ingraham (eingraham2@unl.edu) is an associate professor of art at the University of Nebraska.

Stephen Ramsay (sramsay.unl@gmail.com) is the Susan J. Rosowski Associate Professor of English at the University of Nebraska.

Brian Moore (brian.moore@unl.edu) is associate professor of music education and music technology at the University of Nebraska.

This material is based upon work supported by the National Science Foundation under grant no. 1122956. Additional support was provided by a University of Nebraska-Lincoln (UNL) Phase II Pathways to Interdisciplinary Research Centers grant. Any opinions, findings, conclusions, or recommendations expressed in our materials are solely those of the authors and do not necessarily reflect the views of the National Science Foundation or UNL.