Integrating Computational and Creative Thinking to Improve Learning and Performance in CS1

L.D. Miller, Leen-Kiat Soh, Vlad Chiriacescu
Department of Computer Science and Engineering
University of Nebraska, Lincoln
{lmille, lksoh, vchiriac}@cse.unl.edu

Elizabeth Ingraham
Department of Art and Art History
University of Nebraska, Lincoln
eingraham2@unl.edu

Duane F. Shell
Melissa Patterson Hazley
Department of Educational Psychology
University of Nebraska, Lincoln
dshell2@unl.edu, mpatterson.hazley@gmail.com

ABSTRACT

Our research is based on an innovative approach that integrates computational thinking and creative thinking in CS1 to improve student learning performance. Referencing Epstein's Generativity Theory, we designed and deployed a suite of creative thinking exercises with linkages to concepts in computer science and computational thinking, with the premise that *students can leverage their creative thinking skills to "unlock" their understanding of computational thinking*. In this paper, we focus on our study on differential impacts of the exercises on different student populations. For all students there was a linear "dosage effect" where completion of each additional exercise increased retention of course content. The impacts on course grades, however, were more nuanced. CS majors had a consistent increase for each exercise, while non-majors benefited more from completing at least three exercises. It was also important for freshmen to complete all four exercises. We did find differences between women and men but cannot draw conclusions.

Categories and Subject Descriptors

K.3.2. [**Computer and Education**]: Computer and Information Science Education

General Terms

Measurement, Performance, Design, Experimentation, Human Factors.

Keywords

Computational thinking, creative thinking, CS1.

1. INTRODUCTION

The increased demand for computational thinking (e.g., [23][24]) has led to numerous articles in educational research venues. These articles demonstrate the increasing momentum of research addressing the need for effective education in computational thinking, both for CS and across the broader spectrum of STEM and non-STEM disciplines. This research is diverse, ranging from course specifications to course development, from community building to setting policies, and from teaching and learning to assessment [3][4][9][12][14][24].

SIGCSE'14, March 5–8, 2014, Atlanta, Georgia, USA.
Copyright © 2014 ACM 978-1-4503-2605-6/14/03…$15.00.
http://dx.doi.org/10.1145/2538862.2538940

Adding to the above body of research, we proposed an innovative approach [16] by which we aim to improve the learning of computational thinking by blending it with creative thinking. Creative thinking is thinking patterned in a way that tends to lead to creative results [17]. It is not limited to the arts but is an integral component of human intelligence that can be practiced, encouraged and developed within any context [13][15][22]. Epstein's Generativity Theory breaks creative thinking down to four core competencies: *capturing* novelty, *challenging* established thinking and behavior patterns, *broadening* one's knowledge beyond one's discipline, and *surrounding* oneself with new social and environmental stimuli [8]. Our premise is that by blending computational and creative thinking *students can leverage their creative thinking skills to "unlock" their understanding of computational thinking* [20]. In this way, we should be able to make computational thinking more generally applicable to STEM and non-STEM disciplines where students may have creative thinking skills but lack understanding of computing concepts.

Towards ascertaining the feasibility and understanding the impact of our approach, we designed and deployed four creative thinking exercises during the Fall 2012 semester at the University of Nebraska, Lincoln. Over 200 students in four different introductory CS1 courses took the exercises and the exercises counted as part of their final course grades. Each course was tailored to a different target group (CS majors, engineering majors, combined CS/physical sciences majors, and humanities majors) and so these courses contained a mix of students with differing levels of both computational and creative thinking skills. In a previously reported study, Miller *et al.* [16] found that *exercise completion produced a linear increase in course grades and retention of core computational thinking principles in the four CS1 courses*.

Questions remain, however, about whether there are differential impacts on different student populations that might influence decisions about exercise adoption. First, while we can educate students from non-CS disciplines about using computational thinking to support their own disciplinary creative problem solving, there is a potentially serious problem for expanding computational thinking from CS into other non-CS disciplines. The students in non-CS disciplines come from diverse backgrounds and most are likely to have limited understanding of computing concepts that are used as the basis for computational thinking. Without a basic level of understanding, these students may have a very difficult time developing computational thinking fluency. We need to know whether the creative thinking exercises are benefiting these non-CS majors in the same way that they are CS majors.

Second, the CS1 courses involved have students of different experience, including years in school. Implementing and delivering integrated computational thinking and creative thinking requires understanding of what level of experience is needed to best benefit. While all students in CS1 classes are novices in computer science and computational thinking, freshmen students might benefit less from the addition of creative competencies as they have less general knowledge and less experience in college coursework. Also, the creative competencies involved in the exercises, such as broadening and surrounding, draw on larger interdisciplinary bodies of knowledge and more extensive academic and social experiences. Students having more experience to bring to the exercises may be more able to utilize them effectively. To address this, we need to know if freshmen students who are less widely experienced derive fewer benefits from the exercises than more advanced, upper class students, regardless of their majors.

Finally, we incorporate activities into the exercises that are livelier, more contextual, and more hands-on in their "sensory" aspects. This can potentially help broaden participation in CS by engaging more underrepresented female students, and motivating them to study computer science; however, we need to determine how female students may differentially benefit from the exercises.

Hence, we report in this paper the results of an extended study to examine our central contentions about the benefits of merging computational and creative thinking by determining if there were differential impacts on different student populations. Our research questions were: (1) *Is the effect of creative thinking exercises the same for CS majors and non-CS majors?* (2) *Is the effect of creative thinking exercises different for beginning students (freshmen) than for students with more expertise (non-freshmen)?* (3) *Is the effect of creative thinking exercises different for women and men students?*

2. DESIGN PRINCIPLES

Creative thinking is flexible, imaginative, innovative thinking which draws on all of one's skills and experience [1], involves all of one's senses [22], draws on multiple intelligences [11] and uses a variety of primary thinking tools (such as visualizing, imaging, abstracting, scaling, modeling and analogizing [19]) to produce new outcomes (things or processes) of value [18]. Creative thinking is not an innate talent but instead a process that is an integral component of human intelligence which can be practiced, encouraged and developed within any context [13][15][22].

Based on Generativity Theory, Epstein [6][7][8] has identified the following four core cognitive competencies involved in creative thinking: (1) *Broadening*. The more diverse one's knowledge and skills are the more varied and interesting the possible novel new 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. (2) *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. (3) *Surrounding*. Exposure to multiple, ambiguous situations and stimuli create environments where novel strategies and behaviors may emerge—e.g., looking at things in new ways, interacting with new people, and considering multiple sensory representations. (4) *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.

As made clear in Robinson [18] and Friedman [10], *creative thinking skills are vital not only to be creative designers or artists but to also to work and compete effectively in our complex, interconnected, rapidly changing global society*, we thus see Epstein's core creative thinking competencies are a universally applicable skill set that everyone would be eager to learn and use, just as the case for the universality of computational thinking articulated by Wing [23].

Therefore, in our framework, both computational thinking and creative thinking are viewed as cognitive tools that expand the knowledge and skills that one can apply to a problem. The blending of computational thinking with creative thinking is not conceived as a set of dichotomies, but rather as complementary or symbiotic abilities and approaches. Computational tools expand the knowledge and skills that one has available thereby broadening the array of knowledge that one may bring to a problem. For example, *Challenging* forces computational tools to be used in unanticipated and unusual ways leading to the development of new computational approaches to both old and new problems, *Surrounding* creates new ways of looking at problems and attention to different stimuli and perspectives that may be relevant to how a problem is approached computationally, and *Capturing* forces consideration of new ways to represent and save data and solution procedures.

The principles underlying the design of our creative thinking exercises are (1) balancing of attributes between computational and creative thinking and (2) mapping between computational and creative concepts and skills as they manifest in different disciplines. Table 1 below shows the differing attributes, concepts, and skills that need to be balanced and mapped.

3. EXERCISE DESIGN

All our exercises combine hands-on collaborative problem solving with written analysis and reflection—with sufficient scope and depth such that that each requires creative thinking and teamwork. Each exercise has four common components.

First, the *Objectives* component, at the beginning of the exercise handout, lists the computational and creative thinking objectives for the exercise to help the student understand why they are doing these exercises by showing how an exercise lacking any programming code is related to CS concepts and how students can use relevant creative thinking skills (described in terms of Epstein's core competencies of Surrounding, Capturing, Challenging and Broadening) to solve the exercises.

Second, the majority of the content of an exercise handout is the *Tasks* component which lists all the tasks the group of students must complete during the two weeks of the exercise. These tasks require students to work collaboratively (and may require individual contributions as well.

Third, the *CS Light Bulbs* are additional text snippets highlighting linkages between the exercise tasks and a set of CS topics to help students recognize the associated or practiced computational thinking skills while performing the creative thinking tasks.

Fourth, the *Questions* component uses open-ended questions that require collaboration among students as they engage in both analysis and reflection. To answer these questions, students must apply *creative thinking* to CS problems and revise the results of their original tasks using *computational thinking*. Questions thus build upon the CS Light Bulbs and reinforce the connections between the creative thinking fostered by the tasks and the computational thinking in the CS topics.

Table 1. Balancing attributes and mapping concepts and skills between computational and creative thinking.

BALANCING OF ATTRIBUTES	
Computational Thinking	**Creative Thinking**
Convergent thinking.	Surrounding with new social and environmental stimuli.
Linear and sequential "flow"	Challenging established solutions and algorithms.
Rational & logical processes	Broadening possible solutions through additional paradigms
Methodical	Capturing novelty and spontaneous outputs
MAPPING CONCEPTS & SKILLS	
Computational Thinking	**Creative Thinking**
Algorithmic Thinking – algorithms as models of computational processes*	Challenging accepted solutions and procedures; fostering novel solutions
Programming Fundamentals: (i.e. data models, encapsulation, testing and debugging)*	Broadening applications through new ways of framing and applying fundamental computational knowledge.
Computing Environments – languages & paradigms, tools, applications*	Surrounding with new social and physical environments to broaden perspectives
Data representation: (i.e. data types, variables)*	Capturing new ways of representing/storing data & algorithms.

*CC2001: IEEE/ACM Computing Curriculum, 2001

Although developed separately, our creative thinking exercises are most conceptually similar to the creative thinking activities in the CS4HS project [2]. The two main differences are that our exercises are designed for students, rather than instructors, and cover CS content suitable for college rather than high school.

4. EXERCISE EXAMPLES

Here we describe one of the creative thinking exercises in more detail. We developed four different exercises for the Fall 2012 deployment: (1) Everyday Object where students identified an everyday object (e.g., a hammer, a nail clipper) and described its functionality and I/O as if the object has not been invented, (2) Storytelling where students work on different pieces of a story separately and then come together to resolve all inconsistencies to produce a coherent story, (3) Cipher where student teams developed mapping rules to encode messages and then decode those messages from other teams, and (4) Exploring where students visited a particular location on campus and documented what they sensed. Due to space considerations we only highlight Everyday Object in the following exercise description.

4.1 Objectives

There are two sets of objectives: computational and creative. ***Computational***: (1) Learning about the description and design process for modular programming by describing an everyday object in detail including why the object is needed and how the object functions; (2) Learning about abstraction and function characterization by identifying properties of an everyday object. ***Creative***: (1) Surrounding: looking at an everyday object in new ways, using all of your senses to understand how it's made and how it functions; (2) Capturing: using written language to describe all the different details and characteristics of this everyday object so you can work with it in new ways; (3) Challenging: describing the operations of an everyday object with words and also as a computer program; (4) Broadening: imagining that this object doesn't exist and acting like its inventor who is trying to fulfill a need by creating something new and useful.

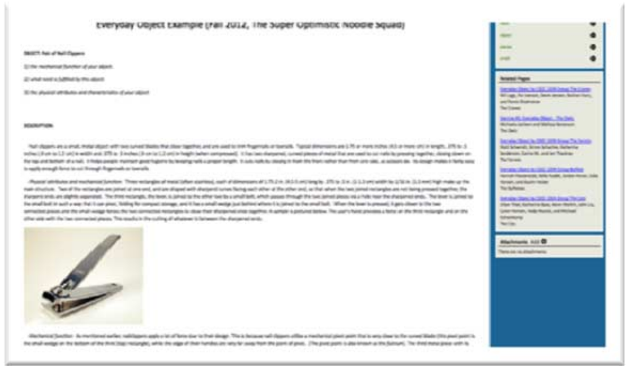
4.2 Tasks

The tasks are divided into two weeks, during which the students will be using language to try to clearly and thoroughly describe the functions of an ordinary object that they might use every day. The students are prompted to act like the inventor of that object, imagining that it does not yet exist and trying to describe what need would be fulfilled by their chosen (new) object and how (specifically) it will function.

Each group will choose a common, everyday object from the list—e.g., zipper, mechanical pencil, binder clip, Ziploc bag, nail clipper, umbrella, can opener, sticky notes, etc. Their challenge is to imagine that this object does not exist and to describe in written language (1) the mechanical function of your object, (2) what need is fulfilled by this object, and (3) the physical attributes and characteristics of their chosen object.

During week 1, each team must describe the object's function, the need it will fulfill and its attributes in clear, non-technical language which any user could understand. Their 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. Week 2 activities involve analysis and reflection on week 1 activities.

Figures 1 shows an example screenshot of student work where the chosen object to be described was a nail clipper.



Everyday Object Example (Fall 2012, The Super Optimistic Noodle Squad)

[illegible]

Figure 1. Screenshot example of student work on the Everyday Object exercise: Object Description.

4.3 CS Light Bulbs

CS Light Bulbs are text snippets highlighting the linkages between the exercise and some CS concepts. Here we provide one example lightbulb. "This description process is very important for developing algorithms in computer science. An algorithm consists of the series of steps necessary to solve a given problem. By using algorithms, we can solve problems without having to constantly "reinvent the wheel" and spend the time, money, etc. to figure out each step ourselves. However, if one or more of these steps are unclear, we can have difficulty following the algorithm which can lead to serious repercussions as described in the following two examples. First, if the formulation algorithm used to mix the concrete for a road or bridge is unclear, workers may make a mistake during pouring leading to reduced service life. Second, if the business plan algorithm for a new company is confusing, venture capitalists may be reluctant to invest leading to failure of the business. To avoid these repercussions, the developer should make every effort to make the algorithm's description as clear as possible for all steps. In other words, characterization of processes is key; it allows us to abstract a process and then convert it into a formal problem or solution."

Note: The handout also includes three other Light Bulbs on writing functions in CS, the diagramming process, and abstraction in programming languages.

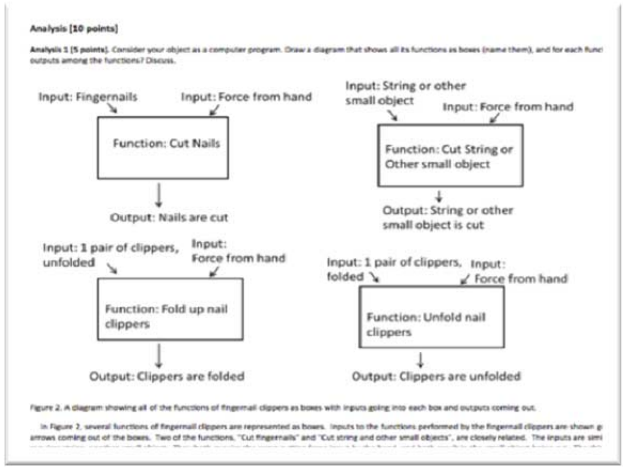
4.4 Questions

For each exercise, there are two sets of questions: Analysis and Reflection. Analysis questions are designed to help them relate to CS concepts as well as to re-examine or revisit the details of the activities. Reflection questions are designed to prompt students to think about the activities at a more abstracted level, and how the lessons learned relate to problem solving in general.

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.

Figures 2 shows another example screenshot of student work in addressing one of the analysis questions.


Analysis [10 points]

Analysis 1 [5 points]. Consider your object as a computer program. Draw a diagram that shows all its functions as boxes (name them), and for each func outputs among the functions? Discuss.

Input: Fingernails
Input: Force from hand
Function: Cut Nails
Output: Nails are cut
Input: String or other small object
Input: Force from hand
Function: Cut String or Other small object
Output: String or other small object is cut
Input: 1 pair of clippers, unfolded
Input: Force from hand
Function: Fold up nail clippers
Output: Clippers are folded
Input: 1 pair of clippers, folded
Input: Force from hand
Function: Unfold nail clippers
Output: Clippers are unfolded

Figure 2. A diagram showing all of the functions of fingernail clippers as boxes with inputs going into each box and outputs coming out.

In Figure 2, several functions of fingernail clippers are represented as boxes. Inputs to the functions performed by the fingernail clippers are shown g arrows coming out of the boxes. Two of the functions, "Cut fingernails" and "Cut string and other small objects", are closely related. The inputs are simi

Figure 2. Screenshot example of student work on the Everyday Object exercise: Analysis Question.

5. DEPLOYMENT AND PLATFORM

The exercises were deployed using the Written Agora system [5]. This is a wiki system designed to facilitate online collaboration between groups of students. The wiki system includes a content page where students can work together on completing the tasks and an online forum where students can discuss, with group members, the responses to the analysis and reflection questions. As the wiki was always online, students could log in and work on the exercises whenever it was convenient. The wiki also kept track of all the revisions so that we could determine which students were contributing to the group.

The exercises represented 3-5% of the final grades depending on the course. After completing the tasks and answering the questions, students in each group were assigned individual grades based on their contributions to the group's wiki page.

6. METHODS

Students voluntarily participated in evaluation data collection which was approved by the University of Nebraska, Lincoln Institutional Review Board. The CS1 courses had 241 students initially enrolled and 196 students who completed the courses. Of those who completed the courses, 150 students (133 male, 17 female; 59 freshmen, 49 sophomore, 32 junior, 13 senior; 3 other) consented to participation in the evaluation and 129 students (114 male; 15 female) consented to the use of their course grades and university grade point average. Samples for specific analyses vary due to missing data and are shown in Tables.

Course grades were used to determine impact on student achievement in the course. To standardize grades across courses, grades were converted to Z-scores within each course. Retention of core computational thinking knowledge and skills was assessed by a test developed by CSCE faculty [21]. The computational thinking knowledge test contained 13 conceptual and problem-solving questions for the core computational thinking content common to all CS1 classes. The coefficient alpha reliability estimate was .76. The computational thinking test was administered on a Web platform (Survey Monkey®) during the last week of classes as part of broader evaluation data collection. Students' cumulative Grade Point Averages (GPA) were obtained from university records adjusted to remove the CS1 course grade.

We used Analysis of Covariance (ANCOVA) to test whether the number of exercises completed was associated with higher course grades and computational thinking test scores. Because of low numbers of participants in some cells, we collapsed 0, 1, and 2 exercises completed into a single group.

7. RESULTS AND DISCUSSIONS

7.1 Results

We included students' cumulative GPA as a covariate in all analyses to statistically control for differences that might be attributable to students' general level of academic ability. Students' cumulative GPA was a significant covariate for course grades in all analyses. Students with higher GPAs earned higher grades indicating that that their achievement in the class generally reflected their overall academic achievement. Students' cumulative GPA, however, was not a significant covariate for the knowledge test in all analyses, indicating that general academic achievement was not related to retention of core computational thinking knowledge and skill.

CS Majors and Non-Majors. For course grades (Table 1), cumulative GPA was a significant covariate ($F(1, 107) = 104.64$, $p < .0001$, partial Eta^2 = .494). With GPA controlled, the number of exercises completed was significantly associated with course grade ($F(2, 107) = 11.27$, $p < .0001$, partial Eta^2 = .174). There was a significant linear trend in planned comparisons ($p < .0001$) from 0-2 to 4 exercises completed. Majoring or intent to major was significantly associated with grades ($F(1, 107) = 3.55$, p =

.002, partial Eta^2 = .089). Students considering or already a CSCE major or minor had higher grades (M = .088) than those not considering a CSCE major or minor (M = .014). The major or minor by exercise interaction also was significant (F(2, 107) = 7.50, p = .001, partial Eta^2 = .123). The interaction can be seen in Table 1. Students who were or were considering a CSCE major or minor followed the general overall linear trend. Students who were not considering a CSCE major or minor had a considerable jump from 0-2 exercises to 3 exercises completed and a much smaller increase from 3 exercises to 4 exercises.

Table 2. Course Grades by Exercises and CS Major

Exercises Completed	Major or Considering			Not Considering		
	M	***SD***	***N***	***M***	***SD***	***N***
0-2	-.462	1.11	18	-.788	.894	22
3	.043	.861	18	.393	.858	14
4	.624	.424	20	.575	.684	22

For the computational thinking knowledge test (Table 2), cumulative GPA was not a significant covariate (F(1, 96) = 0.84, p = .361, partial Eta^2 = .013). With GPA controlled, the number of exercises completed was significantly associated with knowledge test scores (F(2, 96) = 4.01, p = .021, partial Eta^2 = .077). Planned comparisons indicated a significant linear trend (p = .006) from 0-2 to 4 exercises completed. Majoring or intent to major in CSCE was not significant (F(1, 96) = 0.20, p = .652, partial Eta^2 = .002). The intent to major or minor by exercise interaction also was not significant (F(2, 96) = 0.85, p = .430, partial Eta^2 = .017).

Table 3. Knowledge Retention by Exercises and CS Major

Exercises Completed	Major or Considering			Not Considering		
	M	***SD***	***N***	***M***	***SD***	***N***
0-2	5.50	3.29	16	6.35	3.47	20
3	7.44	3.42	16	7.38	3.10	13
4	8.82	1.59	17	7.71	2.61	21

Class Standing. For course grades (Table 3), cumulative GPA was a significant covariate (F(1, 107) = 57.70, p <.0001, partial Eta^2 = .350). With GPA controlled, the number of exercises completed was significantly associated with course grade (F(2, 107) = 13.09, p < .0001, partial Eta^2 = .197). There was a significant linear trend in planned comparisons (p < .0001) from 0-2 to 4 exercises completed. Class standing was significantly associated with grades (F(1, 107) = 4.67, p = .033, partial Eta^2 = .042). Upper class students had higher grades (M = .238) than freshmen (M = -.237). The class standing by exercise interaction also was significant (F(2, 107) = 3.80, p = .026, partial Eta^2 = .066). The interaction can be seen in Table 5. Although there is an overall linear trend, freshmen students do not differ for 0-2 exercises and 3 exercises but increase for 4 exercises and upper class students increase dramatically from 0-2 exercises to 3 exercises, then increase a much smaller amount from 3 exercises to 4 exercises. It appears that freshmen students only showed gains when all exercises were completed; whereas, upper class students gained from completing either three or four exercises.

Table 4. Course Grades by Exercises and Class Standing

Exercises Completed	Freshmen			Upper Classman		
	M	***SD***	***N***	***M***	***SD***	***N***
0-2	-.474	1.043	20	-.808	.951	20
3	.-468	1.026	11	.544	.511	21
4	.282	.696	14	.757	.365	28

For the computational thinking knowledge test (Table 4), cumulative GPA was a not a significant covariate (F(1, 96) = 0.68, p = .411, partial Eta^2 = .007). With GPA controlled, the number of exercises completed was significantly associated with knowledge test scores (F(2, 96) = 3.77, p = .026, partial Eta^2 = .073). Planned comparisons indicated a significant linear trend (p = .007) from 0-2 to 4 exercises completed. Class standing was not significant (F(1, 96) = 2.42, p = .123, partial Eta^2 = .025). The class standing by exercise interaction also was not significant (F(2, 96) = 0.28, p = .750, partial Eta^2 = .006).

Table 5. Knowledge Retention by Exercises & Class Standing

Exercises Completed	Freshmen			Upper Classman		
	M	***SD***	***N***	***M***	***SD***	***N***
0-2	5.24	3.56	17	6.63	3.13	19
3	6.56	3.94	9	7.80	2.87	20
4	7.93	1.59	14	8.38	2.58	24

Gender. For course grades (Table 5), cumulative GPA was a significant covariate (F(1, 107) = 71.49, p <.0001, partial Eta^2 = .401). With GPA controlled, the number of exercises completed was significantly associated with course grade (F(2, 107) = 11.29, p < .0001, partial Eta^2 = .174). There was a significant linear trend in planned comparisons (p < .0001) from 0-2 to 4 exercises completed. Gender was not significant (F(1, 107) = 0.12, p = .721, partial Eta^2 = .001). The gender by exercise interaction also was not significant (F(2, 107) = 2.26, p = .109, Eta^2 = .041).

Table 6. Course Grades by Exercises and Gender

Exercises Completed	Men			Women		
	M	***SD***	***N***	***M***	***SD***	***N***
0-2	-.618	.967	37	-.927	1.583	3
3	.112	.805	25	.497	1.063	7
4	.570	.558	38	.868	.234	4

For the computational thinking knowledge test (Table 6), cumulative GPA was a not a significant covariate (F(1, 96) = 1.28, p = .261, partial Eta^2 = .013). With GPA controlled, the number of exercises completed was not significantly associated with knowledge test scores in the overall model (F(2, 96) = 2.34, p = .102, partial Eta^2 = .047). Planned comparisons, however, indicated a significant linear trend (p = .034) from 0-2 to 4 exercises completed. Gender was not significant (F(1, 96) = 0.84, p = .361, partial Eta^2 = .009). The gender by exercise interaction also was not significant (F(2, 96) = 0.61, p = .548, Eta^2 = .012).

Table 7. Knowledge Retention by Exercises and Gender

	Men			Women		
Exercises Completed	***M***	***SD***	***N***	***M***	***SD***	***N***
0-2	6.06	3.28	33	5.00	5.00	3
3	7.78	3.30	23	6.00	2.68	6
4	8.17	2.29	35	8.67	2.08	3

7.2 Discussions and Implications

Research Question 1. *The creative thinking exercises appeared to affect the achievement of non-CS majors more than CS majors.* Non-CS majors who completed two or fewer exercises received grades well below those of CS majors who completed a similar number of exercises and well below the average for non-CS majors. However, non-CS majors completing three or four exercises received grades well above the average for even CS majors and grades equivalent to or even above CS majors completing a similar number of exercises. *The exercises appeared to affect the retention of core course content equally for both CS majors and non-majors.* These findings support our contention that *the creative thinking exercises can bring CS* *computational concepts to non-CS disciplines and improve non-CS students' understanding of computational thinking.*

Research Question 2. *The creative thinking exercises appeared to affect freshmen and upper class students differently*. Upper class students doing either three or four exercises had higher grades. Freshmen students, however, only had higher grades when doing all four exercises. Although upper class students had higher grades overall, freshmen doing four exercises had grades above the average for all upper class students. *The exercises appeared to affect the retention of core course content in similar ways for both freshmen and upper class students, although upper class students scored higher at all exercise completion levels*. These findings suggest that *while more advanced students may perform better overall and may derive somewhat more benefit from the creative thinking exercises, beginning freshmen students also see improvements in their course achievement and knowledge retention*. Nevertheless, it appears to be important for freshmen students to complete all exercises.

Research Question 3. *There were no differential effects for women and men*. Our analysis of gender was limited by the low number of women students. This makes any findings very tentative. In relation to our contention that the addition of creativity may be especially appealing and beneficial to women, there is some indication in Table 5 that the exercises *may* have been associated with higher course achievement for women. This, however, was not statistically significant.

8. CONCLUSIONS

The findings support our central contention that the incorporation of creative thinking exercises based on Epstein's [8] creative competencies can improve learning of computational thinking. Results expand on the Miller *et al*. [16] findings of a linear "dosage effect" for exercise completion by examining whether there were differences in this effect for CS majors and non-majors and freshmen and upper class students. In relation to retention of core computational thinking knowledge from the courses, there were no differential effects of the exercises. For all students the linear "dosage effect" was present with student completion of each additional exercise increasing retention. For grades, the effects were more nuanced. CS majors had a consistent linear increase for each exercise completed, while non-majors had grade increases only for completing at least three exercises. Upper class students had increases for completing at least three exercises, while freshmen students needed to complete all four exercises before there were grade increases. These results suggest that the "dosage effect" is less strong for grades among some sub-populations. We did not find differences between women and men but cannot draw conclusions because of sample size.

We believe that the exercises impact student achievement and learning because they make students deal with computational principles and skills abstracted from coding. This enhances their ability to connect the computational thinking knowledge to more diverse applications consistent with the Unified Learning Model (ULM) [20]. Also consistent with the ULM, completing exercises provides more retrieval and repetition of course creative thinking content, which will strengthen knowledge connections as indicated in the "dosage effect".

Findings are limited by the implementation in only one semester and four courses. Sample sizes in all analyses preclude broad generalization without follow-up studies, which we currently have in progress. However, our results are encouraging. The merger of computational and creative thinking can be realized in exercises that can be successfully implemented in introductory CS1 courses. Furthermore, these exercises can help students improve their course achievement and learning of computational thinking.

9. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant no. 1122956. Additional support was provided by a UNL Phase II Pathways to Interdisciplinary Research Centers grant.

10. REFERENCES

[1] Andreasen, N. C. 2005. *The Creating Brain: The Neuroscience of Genius*. Dana Press.

[2] Blum, L. and Cortina, T. 2007. CS4HS: An Outreach Program for High School CS Teachers. *SIGCSE* 39, 19-23.

[3] Denning, P. J. 2007. Computing is a Natural Science. *CACM* 49, 33-35.

[4] Denning, P. J. 2009. The Profession of IT beyond Computational Thinking. *CACM* 52, 28-30.

[5] Eck, A. Soh, L-K., and Brassil, C. 2013. Supporting active wiki-based collaboration. In *Proc. of CSCL*, 176-183.

[6] Epstein, R. 1996. Cognition, creativity, and behavior: Selected essays. *Praeger*.

[7] Epstein, R. 2005. Generativity theory and creativity. *Theories of creativity*. Hampton Press.

[8] Epstein, R., Schmidt, S., Warfel, R. 2008. Measuring and Training Creativity Comptencies: Validation of a New Test. *Creativity Research Journal* 20, 7-12.

[9] Fletcher, G. H. L. 2009. Human Computing Skills: Rethinking the K-12 Experience. *CACM* 52, 23-25.

[10] Friedman, T. L. 2006. *The World is Flat: A Brief History of the Twenty-First Century*. Farrar, Straus and Giroux.

[11] Gardner, H. 2007. *Five Minds for the Future*. Harvard Business School Press.

[12] Guzdial, M. 2008. Paving the Way for Computational Thinking. *CACM* 51, 25-27.

[13] Kraft, U. 2005. Unleashing Creativitity. *Scientific American Mind* 16, 19-12.

[14] Lewis, C., Jackson M. H., and Waite, W. M. 2010. Student and Faculty Attitudes and Beliefs about Computer Science. *CACM* 53, 78-85.

[15] Michalko, M. 2001. Cracking Creativity. *Ten Speed Press*.

[16] Miller et al. 2013. Improving Learning of Computational Thinking using Creative Thinking Exercises in CS-1 Computer Science Courses. *Frontiers in Education* 43, 1426-1432.

[17] Perkins, D. N. 1984. Creativity by Design. *Educational Leadership* 42, 18-25.

[18] Robinson, K. 2001. *Out of Our Minds: Learning to be Creative*. Capstone.

[19] Root-Bernstein, R. S. and Root-Bernstein, M. M. 2001. *Sparks of Genius: The Thirteen Thinking Tools of the World's Most Creative People*. Mariner Books.

[20] Shell, D. F., Brooks, D. W., Trainin, G., Wilson, K., Kauffman, D. F., and Herr, L. 2010. *The Unified Learning Model: How Motivational, Cognitive, And Neurobiological Sciences Inform Best Teaching Practices*. Springer.

[21] Shell, D. F., Soh, L-K. 2013. Profiles of motivated self-regulation in college computer science courses: Differences in major versus required non-major courses. *J. Sci. Edu. Tech. Technology*. DOI 10.1007/s10956-013-9437-9.

[22] Tharp, T. 2005. *The Creative Habit: Learn it and Use it for Life*. Simon & Schuster.

[23] Wing, J. 2006. Computational Thinking. *CACM* 49, 33-35.

[24] Wing, J. 2010. Computational Thinking: What and Why. *Link Magazine*