

# A Framework for Computational Thinking across the Curriculum

Ljubomir Perković  
School of Computing  
DePaul University  
Chicago, Illinois 60604  
lperkovic@cs.depaul.edu

Amber Settle  
School of Computing  
DePaul University  
Chicago, Illinois 60604  
asettle@cdm.depaul.edu

Sungsoon Hwang  
Department of Geography  
DePaul University  
Chicago, Illinois 60614  
shwang9@depaul.edu

Joshua Jones  
School of Cinema and  
Interactive Media  
DePaul University  
Chicago, Illinois 60604  
jjones@cim.depaul.edu

## ABSTRACT

We describe a framework for implementing computational thinking in a broad variety of general education courses. The framework is designed to be used by faculty without formal training in information technology in order to understand and integrate computational thinking into their own general education courses. The framework includes examples of computational thinking in a variety of general education courses, as well as sample in-class activities, assignments, and other assessments for the courses. The examples in the different courses are related and differentiated using categories taken from Denning Great Principles of Computing, so that similar types of computational thinking appearing in different contexts are brought together. This aids understanding of the computational thinking found in the courses and provides a template for future work on new course materials. Specific examples of computational thinking in the design category are provided in the context of three distinct courses.

## Categories and Subject Descriptors

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

## General Terms

Design, Standardization

## Keywords

Computational thinking, Great Principles of Computing, General education

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*ITiCSE'10*, June 26–30, 2010, Bilkent, Ankara, Turkey.

Copyright 2010 ACM 978-1-60558-820-9/10/06 ...\$10.00.

## 1. INTRODUCTION

The development of computer technologies and computer science has been largely motivated by a desire to support, extend and amplify the human intellect. In order to make an effective use of computer applications and techniques in his/her field, a person needs to have certain skills. One skill is the ability to use basic computer applications such as an editor and a web or file-system browser; this skill is often described as computer literacy. Another skill is a high level understanding of the workings of a computer system, often defined as computer fluency. While computer literacy and fluency are certainly necessary, neither is sufficient for fully realizing the potential that computing can have in augmenting a person's productivity in their field. The third, critical, skill set is the intellectual and reasoning skills that a professional needs to master in order to apply computational techniques or computer applications to the problems and projects in their field, whether the field is in the arts, sciences, humanities, or social sciences.

This third skill was given the name computational thinking in a 2006 CACM article by Jeannette Wing [8]. Computational thinking is not new, however. Many of its elements are as old as mathematics itself such as Euclid's 500 B.C Greatest Common Divisor algorithm. What is different about the recent attention on computational thinking is the emphasis on explicitly defining what it is and explicitly using it to gain new insights into problems in fields outside of computer science. Wing argues in her seminal article that computational thinking is an emerging basic skill and should become an integral part of education.

Although computational thinking has been defined only recently, there are a number of projects with a focus on refining and understanding computational thinking. Here we describe a project in which we developed a framework for implementing Wing's vision in the context of undergraduate education. Our project focuses on using liberal studies courses – part of the education of the vast majority of undergraduates – as a vehicle for the teaching of computational thinking. In conjunction with a diverse group of faculty we have developed a framework that faculty without formal training in information technology can use to

understand and integrate computational thinking into their liberal studies courses. The framework includes examples of computational thinking in a variety of general education courses, as well as sample in-class activities, assignments, and other assessments for the courses. To aid understanding the framework uses a uniform set of keywords to describe the various instances of computational thinking.

There are other NSF-funded projects with a similar focus on integrating computational thinking into the undergraduate curriculum including “Piloting Pathways for Computational Thinking in a General Education Curriculum” [4], “Computational Thinking Showcase: Computing Concepts Across the Curriculum” [2], “Computing Education in Science Context” [6], “Living In the KnowlEdge Society (LIKES)” [5], and “Renaissance Computing” [7]. Our project is unusual in that it involves a broad group of 18 faculty across diverse disciplines including areas outside of computing and the hard sciences and that it focuses on integrating computational thinking into existing, discipline-specific courses.

## 2. THE FRAMEWORK

In this section we introduce the keywords used in the computational thinking framework, introduce the Liberal Studies Program at DePaul University in which all the courses in the framework are taught, and provide an overview of the computational thinking instances found in the 19 courses that form the backbone of the framework.

### 2.1 Principles of Computing

Computation is a broad term that encompasses different tasks, concepts, and techniques. Similarly, computational thinking involves a broad set of approaches and skills. For this project we found it useful to consider different categories of computational thinking and use these categories to understand computational thinking by distinguishing differences and finding similarities between specific examples in different fields.

The categories we started with in this project are those defined by Denning in his “Great Principles of Computing” project [3]. The goal of Denning’s project is to articulate the fundamental principles of computing. Of particular interest to us is one of Denning’s motivations: “To establish a new relationship with people from other fields by offering computing principles in a language that shows them how to map the principles into their own fields.” He claims that the “principles of computing can be organized into seven categories, each emphasizing a unique perspective on computation.” The Great Principles of Computing, according to Denning, are: computation, communication, coordination, recollection, automation, evaluation, and design. Below is our definition of each principle. Note that the definitions are somewhat different from Denning’s because we are defining them in a context larger than computer science.

Computation is the execution of an algorithm, a process that starts from an initial state containing the algorithm and input data, and goes through a sequence of intermediate states until a final, goal state is reached. Communication is the transmission of information from one process or object to another. Coordination is control (through communication, for example) of the timing of computation at participating processes in order to achieve a certain goal. Recollection is the encoding and organization of data in ways to make it

efficient to search and perform other operations. Automation is the mapping of computation to physical systems that perform them. Evaluation is the statistical, numerical, or experimental analysis, and visualization, of data. Design is the organization (using abstraction, modularization, aggregation, decomposition) of a system, process, object, etc.

### 2.2 The Framework Overview

We briefly outline the current Liberal Studies program at DePaul University. The program consists of two parts. The first part is a Common Core which includes, among others, a first-year, two-course sequence in Mathematical and Technology Literacy, designed to teach “how to apply quantitative reasoning and quantitative information, and to critically evaluate real-world issues and problems using modern information technologies (e.g., spreadsheets, databases, statistical analysis software, search engines, programming algorithms).” [1] This sequence is the basis upon which we think the teaching of computational thinking in context can occur in other courses of the Liberal Studies Program.

The second component of the Liberal Studies Program is made up of courses in six distinct learning domains: Arts and Literature, Philosophical Inquiry, Religious Dimensions, Scientific Inquiry, Self, Society, and the Modern World, and Understanding the Past. Students are required to take 2-3 courses in each domain, but have a choice of at least several dozen courses for each domain.

There are some courses in many of the domains in which we think computational thinking can and should be explicitly taught. In Table 1 we list the 19 courses that were reworked in order to form the backbone of the project. The Liberal Studies Domain in which the course is located is also listed in the table.

Course	Title
<b>Scientific Inquiry</b>	
CSC 233	Codes and Ciphers
CSC 235	Problem Solving
CSC 239	Personal Computing
ECT 250	Internet, Commerce, and Society
ENV 216	Earth System Science
ENV 230	Global Climate Change
ENV 340	Urban Ecology
GEO 241	Geographic Information Systems I
HCI 201	Multimedia and the WWW
IT 130	The Internet and the Web
<b>Arts and Literature</b>	
ANI 201	Animation I
ANI 230	3D Modeling
DC 201	Introduction to Screenwriting
GAM 224	Introduction to Game Design
HAA 130	Principles of European Art
<b>Understanding the Past</b>	
HST 221	Early Russia
HST 250	Origins of the Second World War
<b>First Year Program</b>	
LSP 112	Focal Point Seminar (The Moon)
<b>Honors Program</b>	
HON 207	Introduction to Cognitive Science

Table 1: The project courses

Each of these courses were revised in order to make the computational thinking more explicit. Course materials highlighting one or more topics incorporating computational thinking concepts were developed, learning goals for the computational thinking concepts were formulated, and assessments were written for the learning goals. In Tables 2 and 3 we categorize the examples of computational thinking made explicit in the 19 classes by computing principle.

Course	Auto.	Comm.	Comp.
Scientific Inquiry			
CSC 233			XX
CSC 235			XX
ECT 250	XX		
IT 130		XX	
Arts and Literature			
ANI 201	XX		
ANI 230	XX		XX
First Year Program			
LSP 112			XX
Honors Program			
HON 207		XX	

Table 2: Examples of automation, communication, and computation in the chosen courses

Course	Coor.	Desi.	Eval.	Reco.
Scientific Inquiry				
CSC 233			XX	
CSC 235			XX	
CSC 239			XX	
ENV 216		XX		
ENV 230			XX	
ENV 340			XX	XX
GEO 241		XX		
HCI 201		XX		XX
IT 130		XX		XX
Arts and Literature				
ANI 230		XX	XX	
DC 201		XX		
GAM 224	XX	XX		
HAA 130		XX	XX	
Understanding the Past				
HST 221			XX	
HST 250			XX	
First Year Program				
LSP 112		XX		
Honors Program				
HON 207	XX			

Table 3: Examples of coordination, design, evaluation and recollection in the chosen courses

### 3. COMPUTATIONAL THINKING EXAMPLES

In this section we provide three detailed examples from our framework. Each example covers an example of computational thinking in a given course and includes: (A) the catalog course description, (B) a high-level description of the course and computational thinking concepts covered, (C) the computational thinking learning goal, (D) a case discussion and several guiding questions, and (E) an assessment for the specified learning goal. The examples have been chosen from the same computational thinking category; however, for contrast very different types of courses have been chosen. The first is a course in the Scientific Inquiry Domain that introduces geospatial information processing; the second is an Arts and Literature course about game design; and the third is an animation course that focuses on 3-D modeling for gaming. Viewing three different examples of design in widely disparate contexts provides good insight into the breadth of examples found in our project framework.

#### 3.1 Scientific Inquiry: Geographic Information Systems I

(A) Catalogue description: An introduction to the fundamentals of geospatial information processing. Special topics include spatial data types, map design, and animation. Instruction is accomplished through lectures and hands-on computer lab exercises.

(B) Representing land, water and other geographic features

GEO 241 introduces basic concepts and methods that underlie information systems designed to deal with geographically referenced data. Students will get to understand characteristics of geographic data, and learn to apply GIS methods and tools to display and analyze geographic data. Two types of data models are used to represent spatial entities in GIS: vector and raster. A vector model represents spatial entities based on a set of points (or vertices). A raster model represent spatial entities based on a set of regular grid cells.

(C) CT Learning Goals: Students will be able to understand different ways in which spatial entities are abstracted into data (that is spatial data modeling) and comprehend the advantages and disadvantages of each. (CT category: Design)

(D) Geographic features representations, case discussion: Imagine you're flying over the State of New York. Let's focus on two natural landmarks: the Adirondack Mountains and Lake Ontario. Given your understanding of GIS data models reviewed earlier, consider what data model would be suited to representing spatial properties of the Adirondack Mountains, and Lake Ontario. Spatial properties can be generalized into characteristics specific to point, line, area, and volume in terms of geometry type: location, length, direction, area, shape, and concentration. Now sketch the vector and raster view that represents the boundary of Lake Ontario.

[Q1.] Take a close look at your sketch of the vector view.

In what way would a vector model be limited in representing the boundary of Lake Ontario? Is there any merit that a vector model offers in representing the boundary of Lake Ontario? Repeat all this with the raster view of the boundary of Lake Ontario. Then sketch the vector and raster view that represents the elevation of the Adirondack Mountains.

[Q2.] Take a close look at your sketch of vector view. In what way would a vector model be limited in representing the elevation of the Adirondack Mountains? Is there any merit that a vector model offers in representing the elevation of the Adirondack Mountains? Repeat thus with the raster view.

(E) Assessment: The students' job is to estimate the fraction taken by green spaces in Chicago using two sets of data—DLG (Digital Line Graphs) and DOQ (Digital Orthophoto Quadrangles)—that cover the City of Chicago. DLG is vector data and DOQ is raster data. The fraction of green spaces from DLG will be calculated as the sum of vector areas that constitute green spaces divided by the total land area of Chicago. The fraction of green spaces from DOQ will be calculated as the sum of raster areas that constitute green spaces divided by the total land area of Chicago. Vector areas are the sum of trapezia marked as green spaces. Raster areas are the sum of pixels marked as green spaces. The students will answer the following questions:

- Suppose that it turns out that results (that is the fraction of green spaces from DLG and the fraction of green spaces from DOQ) are quite different. What would explain the difference?
- What data model do you think yields better estimation results? Why do you think so? Be sure to link the respective data model to characteristics of geographic phenomena measured.

- Do you think that getting the estimate of the fraction of green spaces from digital data is any better than alternative methods like a field survey? If so, in what way? If not, in what way? You should answer both questions (that is, why better and worse).

### 3.2 Arts and Literature: Introduction to Game Design

(A) Catalog Description: This course approaches the study of computer games from three directions: first, as examples of media that can be analyzed and critiqued for their thematic elements, formal structure, plot and interactive appreciation; second, as complex software artifacts subject to technological constraints and the product of a labor-intensive design and implementation process; and three as a cultural artifact with behaviors and associations comparable in import to other popular art forms. Student will study the principles of game design and use them both to analyze existing games and to develop their own original game ideas. Students will also learn about the process of game development, starting from the game's narrative concept and moving to consideration of a game's components: the representation of the player, of artifacts, the virtual world that contains them and the interaction between them and the player.

(B) Representation of Game Rules Game rules can be categorized into three types: constitutive, operational, and implicit. Operational rules are the guidelines players require in order to play, such as the rules printed on the box of a board game. Constitutive rules are the underlying logical and mathematical structures in the game. Implicit rules are the "unwritten" rules of the game, such as rules about decorum. Important here are the first two types of rules: constitutive and operational. Two games are considered to be the same if there is a 1-1 relationship between the constitutive rules of the two games, so that if you can find a winning strategy in one game you can use the mapping to find a winning strategy in the other game. At the same time, the operational rules for two structurally identical games can vary significantly. While the operational rules are what make a game enjoyable to play, the constitutive rules are the ones that more experienced players are using when they find winning strategies. Abstracting game rules in different ways is an example of computational thinking; it allows students to see the relationship between different abstractions of rules, the modeling of game behavior, and the underlying structure of a game.

(C) CT Learning Goals: Students will be able to abstract the operational rules of a simple board game to find the underlying constitutive rules for the game and use the constitutive rules to comment on strategies that may exist for the game. (CT category: Design)

(D) Tic-Tac-Toe and 3-to-15, case discussion: Students will consider the well-known rules of Tic-Tac-Toe as well as the rules of 3-to-15 described as follows:

1. Two players alternate turns
2. On your turn, pick a number from 1 to 9. You may not pick a number that has already been picked by either player.
3. The first person to obtain a set of exactly 3 numbers that sum to 15 wins the game

4. If all numbers between 1 and 9 have been chosen and no player has a subset that sums to 15, the game ends in a draw

[Q1.] What strategies exist for Tic-Tac-Toe? Include both strategies for winning and for preventing the other player from winning.

[Q2.] What strategies exist for 3-to-15? Include both strategies for winning and for preventing the other player from winning.

[Q3.] Are the two games the same? Why?

[Q5.] Translate a strategy for Tic-Tac-Toe into a strategy for 3-to-15. For example, describe a blocking strategy for 3-to-15 derived from a blocking strategy for Tic-Tac-Toe.

[Q6.] Which game is easier to play using its operational rules? Why?

(D) Chutes and Ladders, case discussion: The Chutes and Ladders children's board game will be considered. The goal of this activity is to find the set of constitutive rules of Chutes and Ladders.

[Q1.] How can you represent the spinner? How can you represent the player's movement on the board without using a board? How will the "chutes" be represented? Explicitly list all of the constitutive rules that must be included for the "chutes" in the game. How will the "ladders" be represented? Explicitly list all of the constitutive rules that must be included for the "ladders" in the game.

[Q2.] How do you handle the winning condition using this model?

[Q3.] Does the purely constitutive version of Chutes and Ladders have the same feel as the original game?

[Q4.] Are there any strategies that the constitutive rules make clear to you?

(E) Assessment: Develop a set of constitutive rules for the board game Candyland by:

1. Constructing a table that represents the positions on the board for each of the color blocks and picture cards. The table should include a representation for the two shortcuts (gumdrop pass and rainbow trail).
2. Describing a way to randomly produce each of the card combinations from the deck.
3. Describing how each of the 3 penalty spaces (gooey gumdrops, lost in the lollypop woods, and stuck in the molasses swamp) will be handled in your rules.
4. Describing the winning condition for the game.

After you have constructed the constitutive rules, describe any strategies that the rules make clear. If there are no strategies that your constitutive rules illuminate, explain why that is. Is it a property of your representation? Or is it a property of the game?

### 3.3 Arts and Literature: 3-D Modeling

(A) Catalog description: This course covers introductory modeling and texturing techniques required to construct 3D objects and scenes to be used for animation and gaming. Topics to be covered include: scene composition, modeling 3D objects with polygons and smooth surfaces, surface materials and texturing, cameras, lighting and rendering.

(B) Modularization in complex 3D modeling: Because most 3D models are inspired by or have a foundation in the world we are familiar with, there is an inherent complexity in what a 3D modeler is expected to achieve in order to give their models an accepted level of believability. On the other hand, the production time and the computer processing of 3D models require them to be simple, not complex. Techniques such as abstraction, modularization, automation, and randomization are necessary to create realistic models that can be efficiently designed and processed.

(C) CT Learning Goals: Students are able to identify visual patterns in a complex environment or object in order to break it into groups of repetitive modular components. They are then able to use automation and randomization to efficiently design a realistic reconstruction of the environment or model in 3D space. (CT categories: Design – abstraction, Automation, Computation – randomization)

(D) Environment Modeling, case discussion: When one considers the problem a modeler faces when she is required to create a field of grass, it quickly becomes apparent that modeling each individual blade, texturing it, and placing each in a scene would be a completely inefficient use of time and even then may not result in a usable finished model. Using modularization and automation, a 3D modeler would instead create a single blade of grass which would then be duplicated to fill the required area. The “organic” seeming placement, rotation, scale, and relative color of individual blades can then be achieved either by a simple randomization script or (more often than not) some random clicks of the mouse.

Students are required to model an interior warehouse environment out of simple polygon primitives. Along with modeling a simple I-beam skeleton, they must create a believable space by modeling three different kinds of inventory and then by arranging them appropriately within the warehouse.

- [Q1.] What challenges are presented when trying to fill the warehouse space with inventory?
- [Q2.] Describe some ways in which those challenges can be met? What are the advantages and disadvantages to each solution?
- [Q3.] Do the solutions change depending on whether the warehouse is modeled for a film as opposed to a video game?
- [Q4.] What issues arise from duplicating model groups? How can these issues be addressed?
- [Q5.] What types of model attributes can randomness be applied to? Which ones give desirable results?
- [Q6.] Does using an automated script for randomness offer any benefits over creating it by hand? Are there benefits to doing it by hand?

(E) Assessment: Students will be required to turn in a 3D models and a rendered image of a finished warehouse space. The image and the model will be evaluated for the student’s ability to assemble efficient groups of modular components, create the illusion of a complex space through the duplication of modular components, and break the patterns of repetition of warehouse inventory with randomization.

## 4. FUTURE WORK

Assessment of a selection of the materials produced for the 19 courses that form the backbone of this framework is on-going. We wish to evaluate how well the materials are conveying the computational thinking concepts so that the examples and assessments can be further refined. Initial results from the assessment have produced revised materials, but this process will continue until at least June 2010. We are also working with faculty from the University of Illinois at Chicago, Loyola University, and the Illinois Institute of Technology in order to evaluate the potential of this approach to integrating computational thinking into the curriculum at their institutions. A workshop in June 2010 will be conducted in order to bring all participants together to assess the project and consider what the next steps should be. Possible avenues for further work include expanding this approach more broadly at DePaul University, perhaps by making computational thinking a formal part of the Liberal Studies Program, and implementing this approach at other types of institutions, whether with our current partners or other interested universities and colleges.

## 5. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0829671. We also wish to thank Brian Boeck, Elena Boeck, Eugene Beiriger, Xiaowen Fang, Jacob Furst, Chris Goedde, Liam Heneghan, Matt Irvine, Craig Miller, Iyad Kanj, Mark Potosnak, Scott Roberts, Robert Rotenberg, and Marcus Schaefer for their work on the “Computational Thinking Across the Curriculum” project.

## 6. REFERENCES

- [1] Depaul University Liberal Studies Program, November 2009. <http://liberalstudies.depaul.edu/>.
- [2] V. Allan. Computational thinking showcase: Computing concepts across the curriculum, November 2009. <http://digital.cs.usu.edu/~allanv/>.
- [3] P. Denning. Great principles of computing. *Communications of the ACM*, 46(11):15–20, 2003.
- [4] C. Dierbach, H. Hochheiser, and C. Ariza. Piloting pathways for computational thinking in a general education curriculum, November 2009. <http://triton.towson.edu/~compthink/index.html>.
- [5] E. Fox, R. Beck, R. Richardson, W. Chung, E. Carr, C. Evia, W. Fan, S. Sheetz, and C. Zobel. Likes (living in the knowledge society). 38th ACM Technical Symposium on Computer Science Education (SIGCSE), 2008.
- [6] S. Hambrusch, C. Hoffmann, J. Korb, M. Haugan, and A. Hosking. A multidisciplinary approach towards computational thinking for science majors. 39th ACM Technical Symposium on Computer Science Education (SIGCSE), 2009.
- [7] L. Soh, A. Samal, S. Scott, G. Meyer, S. Ramsay, E. Moriyama, B. Moore, D. Shell, and U. Chandra. Renaissance computing: An initiative for promoting student participation in computing, November 2009. <http://cse.unl.edu/renaissance/>.
- [8] J. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, March 2006.