# 1 Computational thinking across the curriculum

In this project we focus 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 order to integrate computational thinking across the liberal studies curriculum, a broad consensus among faculty from diverse areas must be reached. In this project we will work toward achieving this consensus by developing a framework that faculty without formal training in information technology can use to understand and integrate computational thinking into their liberal studies courses. We intend, after the completion of this two-year project, to use the framework to integrate computational thinking into a large variety of courses in DePaul's Liberal Studies program and to disseminate the framework so that other institutions can begin the same process.

# 2 The first year plan: Solving the chicken and egg problem

In order to build the framework, we will need the participation of a large, diverse faculty spanning many academic disciplines. The non-computing faculty, however, will not be familiar with the concept of computational thinking. Without an extensive set of computational thinking examples, best practices, and assessment tools across diverse areas, they may be reluctant to join the project.

In the first year of the project, we will generate the examples, best practices, and assessment tools that we need by first focusing on Liberal Studies courses closer to computing – i.e. courses taught by CDM faculty members – because the computational thinking skills taught in those courses are more explicit and easier to understand. After the first year of the project, when the framework becomes populated with examples, we will move to courses further from computing and taught by faculty outside of CDM. We will use the examples already in the framework to uncover computational thinking skills implicitly taught in these courses and/or discover how computational thinking can be applied in the courses in novel and effective ways.

In the first year of the project, we will leverage the unique nature of CDM where technology unites faculty with training in computer science and faculty whose focus lies outside of computer science. By working together, CDM faculty will enable computational thinking to be accessible to faculty across the university.

# 3 What we expect from you

- You will document and categorize instances of computational thinking in the CDM Liberal Studies course assigned to you.
- You will develop best practices for making computational thinking more explicit in the course, and you will develop ways to teach and use computational thinking in the course.

- You will develop computational thinking learning goals for the course. These goals will be clear, achievable, and measurable.
- You will also develop assessment tools that we will use to evaluate the learning of these computational thinking goals.

Note: The PI and Co-PI, as well as all project participants, should be viewed as a resource to help you complete these tasks.

# 4 What is computational thinking?

In a CACM article[1] Wing argues that computer science has developed a set of "computational thinking" skills that have direct application beyond computer science. She argues that the ideas of abstraction, layering of abstractions, and automation – to name a few – are fundamental computer science concepts that have already yielded new insights in the natural sciences and hard social sciences such as economics. Wing gives specific examples of computational thinking in those fields. She argues that computational thinking is an emerging basic skill that should become an integral part of education together with reading, writing, critical thinking, and problem solving.

This project will start implementing Wing's vision by integrating computational thinking into courses in various areas of the liberal arts curriculum. Wing's article unfortunately does not give examples of computational thinking outside the "hard sciences". Wing's article is also not ideal for explaining computational thinking to professionals outside of computing. Part of the contribution of our project is to uncover examples of computational thinking more broadly and find a way to explain computational thinking to faculty in all academic disciplines.

In order to define computational thinking, broadly speaking and in all academic disciplines, we start with a few points made in a presentation by Pat Phillips [2]. She claims that computational thinking, broadly speaking, is when you ask:

- How difficult is the problem?
- How can it be solved?
- How can I rephrase the question so I can solve it?
- How can I break it down into simpler questions?
- How can information technology be applied to the problem?

[1] *Computational Thinking*, Communications of the ACM, Vol. 49, No. 3, March 2006

[2] http://www.cs.cmu.edu/~CompThink/resources/ct_pat_phillips.ppt

- What computational strategies (such as abstraction, modularization, automation, communication, coordination, visualization tools/techniques) might be employed?

Some of these questions, on their own, are broader than computational thinking. But a subset of them is typically indicative of computational thinking.

What computational thinking is not:

- It's not just more technical details for using computers or specific software.
- It's not thinking like a computer.
- It's not programming (necessarily).
- It doesn't always require a computer.
- It's not yet one more thing to add to your curriculum.

One way to decide whether some process is computational thinking is if it can be described using some key computational thinking terms. If any of these terms is used to describe a process/approach/technique, it is likely that computational thinking is involved. We define these terms:

- *Algorithm (or procedure, function):* An algorithm is a set of rules that describe how to solve a problem. An algorithm may be described as a program, pseudo-code or a less formal step-by-step explanation of how do something.
- *Data (or variable, database, queue):* Data is the information that is part of the problem, as well how it is organized and how it is accessed. Distances between neighboring cities is the data for the problem of computing distances between cities, for example.
- *Abstraction (or conceptualization, modularization):* Abstraction is the pulling out of the important details and the generalization the relationship of details to other situations. Math story problems, like game and virtual world design, lend themselves to this. The important details are abstracted.
- *Search (or query in data mining):* A query is the efficient searching for information. It includes, for example, database and web mining.
- *Sensing & Feedback (event & event handling):* In sensing & feedback computations, the flow of computation consists of executions of event handlers that respond to (sensed, detected) events.
- *Iteration (or automation, loops, recursion):* Iterations involve the repeating of a procedure until a desired goal is reached. In math, iterations occur in long division. In science, it occurs in repeating a step of an experiment, until the desired condition is achieved. In game design and development, it happens in the construction of game versions, from initial prototype to the beta.

- *Systems (communication, coordination):* A system is a group of systems or processes that communicate or coordinate between themselves. Processes could be human beings, communities, viruses, or actual running programs/applications. Systems questions include how groups self-organize with decentralized control, including migrating birds, kids on the playground, social groups, and virtual communities, as well as larger entities like economies and biological systems.

# 5 Computational principles

We do not believe that a long list of examples of computational thinking is the best way to organize the results of our project. In order to categorize the computational thinking skills we uncover, we will start by using a framework developed by Denning in his "Great Principles of Computing" project[3]. The project's goal is to articulate the fundamental principles of computing. Of particular interest to us is one of Denning's motivations: "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:

1. *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.
2. *Communication* is the transmission of information from one process or object to another.
3. *Coordination* is control (through communication, for example) of the timing of computation at participating processes in order to achieve a certain goal.
4. *Recollection* is the encoding and organization of data in ways to make it efficient to search and perform other operations.
5. *Automation* is the mapping of computation to physical systems that perform them.
6. *Evaluation* is the statistical, numerical, or experimental analysis of data.
7. *Design* is the organization (using abstraction, modularization, aggregation, decomposition) of a system, process, object, etc.

Note that the above definitions are mostly ours, not Denning's.

[3] *Great Principles of Computing*, http://cs.gmu.edu/cne/pjd/GP/GP-site/welcome.html

These seven "principles" form a foundation which may be useful to recognize, organize, and categorize instances of computational thinking and build a framework that can translate computational thinking to contexts outside of computer science. In this project, we will use those categories as a start and modify them as needed. For example, abstraction is viewed by Denning as belonging to the design principle, when it could be argued that abstraction is a principle on its own.

## 6 Examples of computational thinking

**In biology:** Interpreting DNA strands as data or code is the classic example of computational thinking that was responsible for a revolution in molecular biology. Abstracting the complexity of organic chemistry to linear strings of 4 characters is a great example of abstraction, a *design* principle according to Denning's categorization. DNA strands encode instructions to be used in the development of the organism and this encoding is an example of data organization and of the *recollection* principle. The genetic mutations can be viewed as a randomized computation (*computation* principle). The cellular growth and interactions between cells may be viewed as an instance of the *coordination* and of the *communication* principle, while the complete organism may be viewed as a complete system which, arguably, can be viewed as an instance of the *design* principle. Making the computational thinking explicit in this course would involve a discussion of abstraction, gene organization as a data structure, genetic mutations as a computation whose goal is to "optimize" the species, and cell interactions using the communication and coordination models. Some of this material can refer to topics in technological literacy courses (ISP120/121). Assessment of computational thinking would involve assessing whether students can apply abstraction, computation or other computing principles in the context of the biology class.

**In history:** In a "Crime and Punishment" history topics class, students might have access to a publicly available historical crime statistics database. An interesting final project would be to ask student uncover patterns in the data and find interesting correlations. Searching for the required data in the database would be an example of the *recollection* principle, while the *computation* and *evaluation* principles are illustrated by the formulation of the database queries and the statistical computation, respectively. Making these principles explicit in the class would involve a high-level discussion of databases, queries and basic statistics. The actual technical aspects of these three subjects should NOT be covered, because this is a history class. Instead, references to material in ISP 120 and especially ISP 121 should be made. The computational thinking learning goals for the class could be the understanding of data storage and processing and statistical analysis, and the assessment would involve a few questions asking how to obtain a certain information or correlation from a given database.

**In economics or an ethics of cinema class:** A similar approach could be used in a sociology or economics class to study relationships between popular culture and crime. One could use crime statistics databases and movie databases and study the correlation between

the release of a violent movie (or computer game) and actual violence on a given weekend. (This has actually been done in a recent study[4], and a negative correlation has been found). The computational thinking principles, teaching strategies, learning goals, and assessment tools discussed in the previous example apply here as well. In another example, economies can be modeled as evolving systems of autonomous interacting agents (sensing & feedback illustrating the *computation* principle).

**In medicine:** After making a hypothesis (*abstraction* principle) about the methods of transmission of a disease (person to person, airborne, foodborne or a combination of these) and assuming a certain probability of transmission for each method, one can run simulations (*evaluation* principle) to study how a disease would spread across the population [5]. The results could then be confirmed by actual transmission data. In the context of medicines, medical students will need to grasp the effects of using multiple medicines or off the effects of multiple diseases (*coordination* principle).

**In art:** In order to construct an origami representation of an object, the object's features may be abstracted using graph theory (*design* principle). Once an abstract graph model is obtained, algorithms that construct the sequence of foldings (*computation* principle) have been developed to construct the origami representation of the object. This in turns implies that the origami construction can be automated (*automation* principle). Abstraction, computation and automation are principles that can be explicit when describing other artistic projects that benefit from the use of technology.

**In interactive media:** Media content often includes interacting objects. For example, in computer games, avatars and various objects interact with each other. These objects and interactions can be described as objects (classes) and attributes (methods). (*design* principle). Interactive media can be viewed as a reactive system of objects and/or avatars, one that responds to user generated events [6] (sensing & feedback illustrating the *computation* principle).

## 7 What we expect from you, one more time

- You will document and categorize instances of computational thinking in the CDM Liberal Studies course assigned to you.
- You will develop strategies for making the computational thinking more explicit in the course, so it is something students are aware of. In other words, you will develop ways to teach and use computational thinking in the course, such as sample lectures, in-class activities, and/or specific course assignments.

[4]"Does Movie Violence Increase Violent Crime?", Gordon Dahl and Stefano DellaViqna, *National Bureau of Economic Research*, NBER Working Paper 13718, http://www.nber.org/papers/w13718

[5]http://www.explorelearning.com/index.cfm?method=cResource.dspView&ResourceID=379&ClassID=954421

[6]http://www.cs.cmu.edu/ CompThink/seminars/golan/index.html

- You will develop computational thinking learning goals for the course. These goals will be clear, achievable, and measurable.
- You will also develop assessment tools that we will use to evaluate the learning of these computational goals.
- You will do the assessment, collect the results of the assessment, and report the results back to all project participants.

Note again: The PI and Co-PI, as well as all project participants, should be viewed as a resource to help you complete these task.

The deadline for the final, completed document containing the above materials is February 15, 2009. A first draft of the document is due by January 5, 2009. The PI and Co-PI are available to help you with your draft throughout the Fall quarter 2008 and especially during the December break.

## 8 What is your assigned course?

| | |
|---|---|
| Kanj | CSC 235: Problem Solving |
| Schaefer | CSC 233: Codes and Ciphers |
| Fang | ECT 250: Internet, Commerce, and Society |
| Furst | CSC 239: Personal Computing |
| Miller | HCI 201: Multimedia and the World Wide Web |
| Roberts | ANI 101: Animation for Non-Majors |
| Irvine | DC 105: Digital Media Literacies |
| Jones | ANI 230: 3d Modeling for Animation and Gaming |
| Settle | GAM 224: Introduction to Game Design |
| Perkovic | IT 130: The Internet and the Web |

## 9 Full project management plan

We summarize the project activities, some of which will not concern you directly. Note that the project plan is subject to change, especially during the second year.

| Project milestones | Expected outcomes | Responsible parties |
|---|---|---|
| **October 15, 2008** | | |
| Final selection of CDM Liberal Studies courses | | All CDM participants |
| **October 15, 2008 - January 5, 2009** | | |
| CDM liberal courses analyzed | An analysis of the computational thinking content in the selected courses, their categorization into the principles categories, development of computational thinking learning goals and assessment tools. | All CDM participants |
| **January 5, 2009** | | |
| First draft of CDM liberal studies courses reports | The first draft of an analysis of the computational thinking content in the selected CDM courses. | All CDM participants |
| **January 5, 2009 - February 15, 2009** | | |
| Refinement of CDM liberal studies courses reports | Refinement of the computational thinking content in the selected CDM courses. | All CDM participants |
| **February 15, 2009** | | |
| Final draft of CDM liberal studies courses reports | The final draft of an analysis of the computational thinking content in the selected CDM courses. | All CDM participants |
| **February 15, 2009 - March 27, 2009** | | |
| Preliminary Framework | A preliminary framework document will be developed and shared with interested non-CDM DePaul faculty. | PI, Co-PI, and Denning |
| **June 1, 2009** | | |
| Selection of non-CDM faculty and courses | | PI and Co-PI |
| **March 30, 2009 - June 12, 2009** | | |
| First Assessment | All selected CDM courses taught in the Spring quarter will be assessed. | All CDM participants |
| **July 1, 2009** | | |
| First Assessment Report | A report summarizing the assessment of all CDM courses taught in the Spring quarter will be completed. | PI and Co-PI |

| Project milestones | Expected outcomes | Responsible parties |
|---|---|---|
| **September - November 2009** | | |
| Second Assessment | All selected CDM courses taught in the Fall quarter will be assessed. | All CDM participants |
| Selected non-CDM Liberal Studies courses analyzed | The selected non-CDM courses will be analyzed. The CDM participants will work with the Co-PIs to educate the non-CDM faculty about instance of computational thinking across disciplines. The computational thinking content in the selected courses will be categorized, and computational thinking learning goals and assessment tools will be developed. | All participants (CDM and non-CDM) |
| **December 1, 2009** | | |
| Second Assessment Report | A report summarizing the assessment of all CDM courses taught in the Spring and Fall quarters will be completed. | PI and Co-PI |
| **January - March 2010** | | |
| Third Assessment | All selected CDM and non-CDM courses taught in the Winter quarter will be assessed. | All participants (CDM and non-CDM) |
| **April 1, 2010** | | |
| Invitations for workshop | Invitations for the June workshop will be issued. All participants in the project are expected to participate in the workshop. Other faculty from each participating institution will also be invited, as will interested faculty from other Chicago-area institutions. | PI and Co-PI |
| Third Assessment Report | A report summarizing the assessment of all courses taught in the Spring, Fall, and Winter quarters will be completed. Separate sections for CDM and non-CDM courses will be included. | PI and Co-PI |

| Project milestones | Expected outcomes | Responsible parties |
|---|---|---|
| **April - June 2010** | | |
| Fourth Assessment | All selected CDM and non-CDM courses taught in the Spring quarter will be assessed. | All participants (CDM and non-CDM) |
| **June 15, 2010** | | |
| Workshop | The workshop will involve all project participants and other interested regional faculty. The main purpose of the workshop will be to review, discuss, and revise the documents created so far with a particular focus on the framework document and assessment tools. | All project participants and other interested faculty |
| **July 1, 2010** | | |
| Fourth Assessment Report | A report summarizing the assessment of all courses taught in the Spring, Fall, and Winter quarters will be completed. Separate sections for CDM and non-CDM courses will be included. | PI and Co-PI |
| Framework Finalized | A final framework document will be developed. | PI and Co-PI with Denning |