Class 54: What is Computer Science? Revisited

Held: Tuesday, 8 December 2009

Summary: We begin to conclude our introduction to CS by looking beyond this course to what kinds of topics computer scientists often study.

Related Pages:

* EBoard.
* Due: Exam 3: Types and Algorithms.

Notes:

* Tomorrow is our "What did I learn this semester?" day.
* Please fill out the post-course survey at
  `http://www.grinnell.edu/academic/psychology/faculty/dl/risc/.`

Overview:

* What is CS?
* Subfields of CS.
* Related Disciplines.

What is CS?

* We started the course with this question.
* We therefore start to end the class with a similar question.
* I've given you a simple definition: *Computer science is the study of algorithms and data.*
* We've seen a bit what an algorithm is and what data are.
* But how do we study them?
* How have we studied them in this course?

A Professional Discussion

* In late April and early May 2008, the SIGCSE (Special Interest Group in Computer Science Education) listserve had an interesting discussion about the definition of CS, spurred, in part, by a comment on the role of Calculus in CS and ABET's decision to remove the Calculus requirement.
  * ABET is the engineering accrediting board
* I particularly like the following message.

From: Michael Feldman Subject: Re: [SIGCSE-members] CS definitions Date: May 3, 2008 6:18:28 PM CDT To: SIGCSE-members

Computing Curricula 2005 was developed by a joint task force of members from The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), and The IEEE Computer Society (IEEE-CS).

Here's the definition of Computer Science that appears in CC 2005, in the overview document at http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf:

"Computer science spans a wide range, from its theoretical and algorithmic foundations to cutting-edge developments in robotics, computer vision, intelligent systems, bioinformatics, and other exciting areas.

"We can think of the work of computer scientists as falling into three categories.

"-- They design and implement software. Computer scientists take on challenging programming jobs. They also supervise other programmers, keeping them aware of new approaches.

"-- They devise new ways to use computers. Progress in the CS areas of networking, database, and human-computer-interface enabled the development of the World Wide Web. Now CS researchers are working with scientists from other fields to make robots become practical and intelligent aides, to use databases to create new knowledge, and to use computers to help decipher the secrets of our DNA.

"-- They develop effective ways to solve computing problems. For example, computer scientists develop the best possible ways to store information in databases, send data over networks, and display complex images. Their theoretical background allows them to determine the best performance possible, and tstudy of algorithms helps them to develop new approaches that provide better performance."

I also think the CSAB definition at http://www.csab.org/comp_sci_profession.html bears repeating:

"Computer science is a discipline that involves the understanding and design of computers and computational processes. In its most general form it is concerned with the understanding of information transfer and transformation. Particular interest is placed on making processes efficient and endowing them with some form of intelligence. The discipline ranges from theoretical studies of algorithms to practical problems of implementation in terms of computational hardware and software. A central focus is on processes for handling and manipulating information. Thus, the discipline spans both advancing the fundamental understanding of algorithms and information processes in general as well as the practical design of efficient reliable software and hardware to meet given specifications. Computer science is a young discipline that is evolving rapidly from its beginnings in the 1940's. As such it includes theoretical studies, experimental methods, and engineering design all in one discipline. This differs radically from most physical sciences that separate the understanding and advancement of the science from the applications of the science in fields of engineering design and implementation. In computer science there is an inherent intermingling of the theoretical concepts of computability and algorithmic efficiency with the modern practical advancements in electronics that continue to stimulate advances in the discipline. It is this close interaction of the theoretical and design aspects of the field that binds them together into a single discipline.

"Because of the rapid evolution it is difficult to provide a complete list of computer science areas. Yet it is clear that some of the crucial areas are theory, algorithms and data structures, programming methodology and languages, and computer elements and architecture. Other areas include software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, computer-human interaction, computer graphics, operating systems, and numerical and symbolic computation.

"A professional computer scientist must have a firm foundation in the crucial areas of the field and will most likely have an in-depth knowledge in one or more of the other areas of the discipline, depending upon the person's particular area of practice. Thus, a well educated computer scientist should be able to apply the fundamental concepts and techniques of computation, algorithms, and computer design to a specific design problem. The work includes detailing of specifications, analysis of the problem, and provides a design that functions as desired, has satisfactory performance, is reliable and maintainable, and meets desired cost criteria. Clearly, the computer scientist must not only have sufficient training in the computer science areas to be able to accomplish such tasks, but must also have a firm understanding in areas of mathematics and science, as well as a broad education in liberal studies to provide a basis for understanding the societal implications of the work being performed."

Let me pull out and emphasize one part of the 3rd paragraph, because (in my opinion) it's really the nub of the profession and consequently of our responsibility as educators:

"[A] well educated computer scientist should be able to apply the fundamental concepts and techniques of computation, algorithms, and computer design to a specific design problem. The work includes detailing of specifications, analysis of the problem, and provides a design that functions as desired, has satisfactory performance, is reliable and maintainable, and meets desired cost criteria."

If that's not a clear, crisp, understandable, effective definition of our profession, I don't know what is.

Mike Feldman

Subfields of Computer Science

- In part, the ways in which we study algorithms lead to different subfields of computer science. The domains we study also lead to different classes of algorithms.
- Computer scientists who emphasize *organization and architecture* study the ways in which algorithms and data may be implemented in hardware and the implications of particular implementations.
  - Example: Representing integers and adding integers.
  - Grinnell's course: CSC 211
- Computer scientists who emphasize *operating systems* study algorithms and data representations that permit programs to use the common resources (file system, processor, mouse, etc.)
  - Example: Dining philosophers problem.
  - Grinnell's course: CSC 213
- Computer scientists who emphasize *software engineering* look at the design of large computer applications. They may consider process, program segmentation, team aspects, or even social aspects.
  - Example: Waterfall vs. Agile

- ○ Grinnell's courses: CSC 323/335
- Computer scientists who emphasize *artificial intelligence* look for ways to either model the way the brain works or to build alternate simulations.
  - ○ Example: Genetic programming.
  - ○ Grinnell's course: CSC 261
- Computer scientists who emphasize *algorithms* tend to look for interesting problems for which to design algorithms. They also investigate general algorithm design strategies and prove things about the characteristics of problems and algorithms.
  - ○ Example: Lower-bound on the running time of a sorting algorithm.
  - ○ Example: The traveling salesman problem.
  - ○ Grinnell's' course: CSC 301
- Computer scientists who emphasize *programming languages* look at the design of the languages in which we express algorithms.
  - ○ Example: Paradigms
  - ○ Grinnell's course: CSC 302
  - ○ CSC 362 considers who we *implement* these languages.
- Computer scientists who emphasize *theory* consider models of computation and the limits of these models.
  - ○ Example: The halting problem.
  - ○ Example: The Church-Turing thesis.
  - ○ Grinnell's course: CSC 341
- Computer scientists who emphasize *computer graphics* write algorithms related to that problem domain.
  - ○ Example: Ray tracing.
  - ○ Grinnell's course: Special Topics
- Computer scientists who emphasize *human-computer interaction* consider the relationship of the programs we write to the people who use them.
  - ○ Example: Menu ordering.
  - ○ Grinnell's course: Special Topics
- Computer scientists who emphasize *databases* look carefully at the models for representing and accessing large amounts of data.
  - ○ Example: Relations
  - ○ Grinnell's course: Special Topics
- Computer scientists across the board consider *social and societal implications of computing*, although some focus more on this area than others. (And yes, some make it their specialization, and we still consider them computer scientists.)
- And that's just some of them.

Related Disciplines

- There are many fields that are like computer science, but differ a bit.
- Here are some variants that you may hear.
- *Computer Engineering* emphasizes the construction of computational devices.
- *Software Engineering* (mentioned above) emphasizes the construction of software, particularly

processes, using some of the approaches of the discipline of engineering.

- *Informatics* is either (a) a better, European, name for CS or (b) the application of CS to a particular problem domain.
- *Information Science* is the study of information. It tends to focus more on representation or the softer sides of CS.
- *Computer Programming* is a profession not always closely related to CS.