Computational Thinking

Related Efforts

CS Principles – Big Ideas

- Computing is a creative human activity that engenders innovation and promotes exploration.
- Abstraction reduces information and detail to focus on concepts relevant to understanding and solving problems.
- Data and information facilitate the creation of knowledge.
- Algorithms are tools for developing and expressing solutions to computational problems.
- Programming is a creative process that produces computational artifacts.
- Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems.
- Computing enables innovation in other fields including science, social science, humanities, arts, medicine, engineering, and business.

CS Principles - Practices

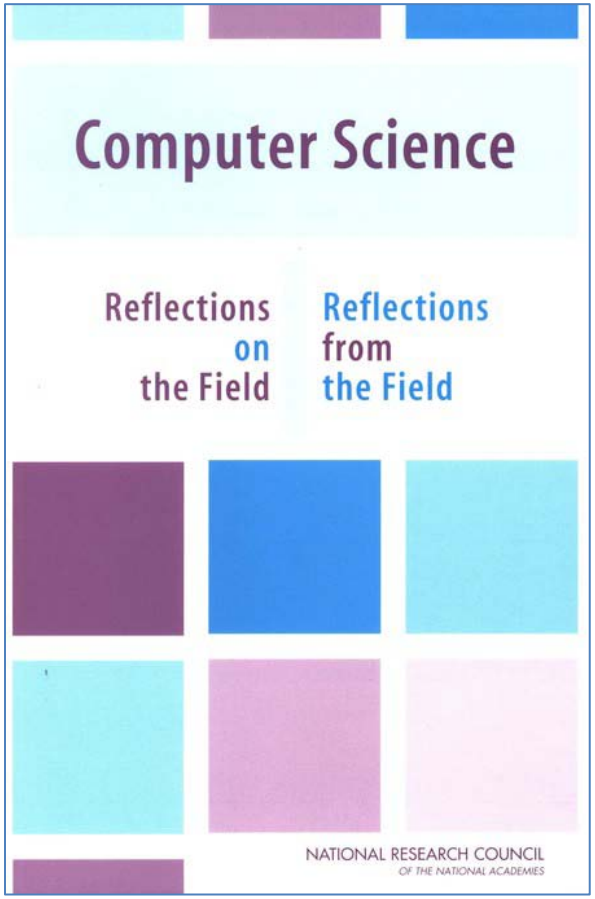
- Analyzing problems, artifacts, and effects of computation
- Creating and using computational artifacts, computational models
- Communicating processes and results
- Connecting computation with mathematics, science, engineering
- Work effectively in teams

Ed Fox Categories

- Derived from experience with LIKES project
- What others want from CS
  - Modeling and simulation (prediction, analysis,…)
  - Representation (data, knowledge,…)
  - Interaction (HCI, VR, graphics,…)
  - Algorithms (workflows, procedures,…)

A Perspective on Computer Science

- Sponsored by National Research Council
- Focuses on research areas
- "…the discussion that follows does not aim to explicitly or comprehensively define computer science or to catalog all of the research areas. Instead, the approach is to indicate and illustrate the essential character of the field through a sampling of representative topics." [p 11-12]

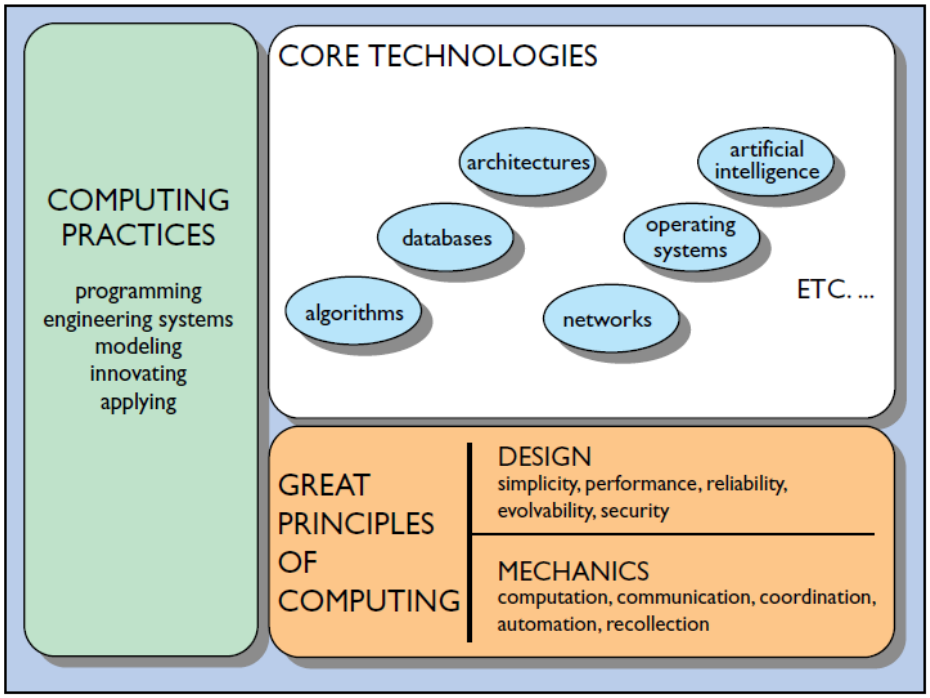


Representative Topics in CS

- Computer Science Research [NRC 2004]
  - Involves **Symbols** and Their Manipulation,
  - Involves the Creation and Manipulation of **Abstraction**
  - Creates and Studies **Algorithms**
  - Creates **Artificial Constructs**, Notably Unlimited by Physical Laws
  - Exploits and Addresses **Exponential Growth**
  - Seeks the **Fundamental Limits** on What Can Be Computed
  - Often Focuses on the **Complex, Analytic, Rational Action That Is Associated with Human Intelligence**

Great Principles

- Computing mechanics
  - The laws governing computations
- Design principles
  - Conventions for designing computations
- Computing practices
  - Standards for implementation
- Core technologies
  - Shared attributes of application domains



Justification questions

- Each major element has a characteristic question that justifies its place in the hierarchy and exposes the integral role of practice.

Levels	Central Questions	Example Technologies
Application Domains	How do we work with others to design computing that serves them?	Supercomputers, grid computing, domain databases, graphics design, interfaces, ...
Core Technologies	How do we design computations that support common elements across applications?	Algorithms, databases, networks, operating systems, HCI, AI, …
Design Principles	How do we organize ourselves to build computations that work?	Design tools, object-oriented programming, layering, virtual machines, authentication, …
Computing Mechanics	How do computations work?	Logic simulators, protocol stack, workflow, expert systems, virtual memory, …

Great Principles

- "The principles of a field are actually a set of interwoven stories about the structure and behavior of field elements." [Denning 2003]
- Computer Science is seen as Computing Mechanics in parallel to the use of the term "mechanics" in other disciplines
- "Computing Mechanics deals with the structure and operation of computations. It does so with stories … [that] I could group … into the five categories." [Denning 2003]
- It is "better to view the categories as windows into computing mechanics." [Denning 2003]

Great Principles: Categories

Window	Central Concern	Principal Stories
Computation	What can be computed; limits of computing.	Algorithm, control structures, data structures, automata, languages, Turing machines, universal computers, Turing complexity, Chaitin complexity, self-reference, predicate logic, approximations, heuristics, non-computability, translations, physical realizations.
Communication	Sending messages from one point to another.	Data transmission, Shannon entropy, encoding to medium, channel capacity, noise suppression, file compression, cryptography, reconfigurable packet networks, end-to-end error checking.
Coordination	Multiple entities cooperating toward a single result.	Human-to-human (action loops, workflows as supported by communicating computers), human-computer (interface, input, output, response time); computer-computer (synchronizations, races, deadlock, serializability, atomic actions).
Automation	Performing cognitive tasks by computer.	Simulation of cognitive tasks, philosophical distinctions about automation, expertise and expert systems, enhancement of intelligence, Turing tests, machine learning and recognition, bionics.
Recollection	Storing and retrieving information.	Hierarchies of storage, locality of reference, caching, address space and mapping, naming, sharing, thrashing, searching, retrieval by name, retrieval by content.

Great Principles – Design and Practice

- Design
  - "Computing professionals follow principles of design that enable them to harness mechanics in the service of users and customers."
  - Principles
    - Simplicity
    - Performance
    - Reliability
    - Evolvability
    - Security
- Practices
  - Programming
  - Engineering of systems
  - Modeling and Validation
  - Innovating
  - Applying

Denning's Criticism

- Premise (possibly misplaced?)
  - "Computational thinking is seen by its adherents as a novel way to say what the core of the field is about, a lever to reverse the decline of enrollments, and a rationale for accepting computer science as a legitimate field of science." [Denning 09]
- Questions (are these fair questions?) [Denning 09]
  - "Is computational thinking a unique and distinctive characterization of computer science?"
  - "Is computational thinking an adequate characterization of computer science?"

Denning's Criticism

- What is CT?
  - Equates with "algorithmic thinking"
  - "…it means a mental orientation to formulating problems as conversions of some input to an output and looking for algorithms to perform the conversion."
  - Expanded to include:
    - Multiple levels of abstraction
    - Algorithms developed using mathematics
    - Scalability concerns

Denning's Criticism

- Relates computational thinking to computational science
  - "…computational science is seen in the other sciences not as a notion that flows out of computer science, but as a notion that flows from science itself."
  - "Computational thinking is seen as a characteristic of this way of science. It is not seen as a distinctive feature of computer science."
- Does this misrepresent computational thinking?
- His conclusion:
  - "Computation is widely accepted as a lens for looking at the world. We do not need to sell that idea. Computational thinking is one of the key practices of computer science. But it is not unique to computing and is not adequate to portray the whole of the field."

Interesting Observations

- "Computation is present in nature even when scientists are not observing it or thinking about it. Computation is more fundamental than computational thinking."
- "The great principles framework reveals that there is something even more fundamental than an algorithm: the representation. Representations convey information. A computation is an evolving representation and an algorithm is a representation of a method to control the evolution."
- In the great principles framework "… computational thinking is not a principle; it is a practice. A practice is a way of doing things at which we can develop various levels of skill."

References

- *[CSP] CS Principles. http://www.csprinciples.org/*
- *[NRC 2004] Computer Science: Reflections on the Field, Reflections from the Field*. 2004, National Research Council.
- [Denning 2003] Peter Denning, Great Principles of Computing, Communications of the ACM , 46(11), November 2003, p. 15-20.
- [Denning 2009] Peter Denning, Beyond Computational Thinking, Communications of the ACM , 52(6), June 2009, p. 28-30.