CRA-E (Computing Research Association – Education) White Paper

"My question is: Are we making an impact?"

Contributors

- Committee
  - Andy van Dam, Brown (chair)
  - Jim Foley, Georgia Tech
  - John Guttag, MIT
  - Pat Hanrahan, Stanford
  - Chris Johnson, University of Utah
  - Randy Katz, UC Berkeley
  - Henry Kelly, DOE (formerly FAS)
  - Peter Lee, DARPA and CMU
  - David Shaw, D.E. Shaw Research
- Support
  - Andrew Bernat, CRA Executive Director
  - Rosemary Michelle Simpson, Brown (Editor)
- Substantive reviews
  - Jeannette Wing, Lynn Andrea Stein, and Ed Fox

Context

- Causes for concern
  - statistics and trends, pipeline issues (workforce, research careers)
    - CS AP course
    - undergraduate enrollments
    - diversity issues
  - perception issues – how to recapture the magic?
  - recognition issues
    - NAS science education study currently out for public review covers
      - life, earth & space, and physical sciences, and engineering & technology
      - but not a word about Computational Thinking – purely computer as a tool, like a microscope
- Many universities/departments are restructuring to adapt to changing conditions

Related Efforts

- ACM's CSTA Model Curriculum for K-12 Computer Science (2003)
- NSF-triggered CS/10K project - K-12 (in progress)

CS 10K

GOAL: Develop an effective high school curriculum and get it taught by in 10K schools by well-prepared teachers by 2015.

Curriculum based around new AP course: *CS: Principles*

- AP is the only point of national leverage, rigorous, popular with students and schools, fidelity of replication
- *CS: Principles* covers fundamentals of computing & is engaging and inspiring. It's being piloted at 5 colleges 2010-11 with more to follow 2011-2012

Deployment needs assessments, pre-service & in-service teacher training, ongoing professional development, and gain entrée into the schools

We'll need the computing community's help!

www.csprinciples.org

Related Efforts

- ACM's CSTA Model Curriculum for K-12 Computer Science (2003)
- NSF-triggered CS/10K project - K-12 (in progress)
- Denning's Great Principles and Rebooting Computing: the Magic and Beauty of Computer Science (2009)
- CSTB "Report of a Workshop on The Scope and Nature of Computational Thinking" (2009)

Mission Statement

CRA-E's mission is to explore the issues of undergraduate education in computing and computational thinking for those who will do research in disciplines from the sciences to the humanities.

As technology and teaching methodologies continue to evolve, how should programs in computer science, computational science, and information science co-evolve?

Can we communicate a core set of ideas, principles, and methodologies that is domain-independent?

Mission Boundaries

- Not part of our charter
  - K-12
  - preparing undergraduates for careers in general
  - curriculum design
- Focus
  - preparing undergraduates for computationally-oriented research careers
  - environment design
- We wanted to but weren't able to consider co-evolving technology and pedagogy
  - "A teacher for every learner"
    - "Grand Challenge 3. Provide a Teacher for Every Learner" in Grand Research Challenges in Information Systems workshop sponsored by CRA

Goal of White Paper

- Provide guidance enabling institutions to
  - create an undergraduate environment that
  - supports acquisition and internalization of the computationally-oriented researcher mindset.
- Two sub-goals
  - identify issues facing faculty charged with educating computationally-oriented researchers in all fields in the first part of the 21st century
  - make recommendations that are relevant and implementable within the current institutional context
- Target audience
  - university and college faculty, school chairs, deans, provosts, as well as government policy-makers, and

Method of Operation

- Leverage best practices
  - e.g. CMU, Cornell, Georgia Tech, Stanford
- Leverage best writings
  - e.g. Jeannette Wing, Lynn Andrea Stein, Peter Denning
- Focus on environment design
  - not curriculum design
  - focused on undergraduate education
    - others are dealing with K-12, job training…

Categories of Recommendatons

- Introduce students to computational thinking
  - foundational courses that address their interests
  - persistent concepts and skills
- Refactor CS curricula
  - lean core plus flexible and adaptable set of options
  - address future computationally-oriented directions
- Identify cognitive, mastery, and research skills
  - pervade the entire curriculum from introductory courses through advanced senior-level courses

CRA-E Recommendations

- Computationally-Oriented Foundations
  1. Introductory Courses
     addressing a broad range of student interests
- Refactoring Computer Science Curricula
  2. Core/Foundation for All CS Graduates
     lean core with focus on enduring concepts, techniques, and skills
  3. Specialization: Tracks, Threads, and Vectors
     flexible approaches to gaining deeper understanding and skills
  4. Specialization: Integrated Joint Majors
     deep collaboration among disciplines
- Develop Mastery across the Curricula
  5. Design Under Constraints and the Gaining of Mastery
     deepen the skill set
  6. Prepare Students for Research Careers
     develop computationally-oriented researchers

1. Introductory Courses

- Problem
  - how do we address a diverse undergraduate population to these introductory courses?
  - what cognitive skills, concepts, and techniques should be included in these courses?
  - how do we leverage the personal interests and abilities of students?
- Approach
  - provide range of introductory courses that address the interests of a broad range of students
    - contextualized computing
    - theoretical/abstraction-oriented
    - traditional CS intro programming
  - experiment with non-traditional approaches such as building robots, working with collecting, analyzing, and visualizing

Working Definitions

- Cognitive skills
  - A cognitive skill is a mental skill required to understand and practice computational subjects.
- Concepts
  - A concept is a named abstraction that has a definition, such as recursion and concurrency
- Techniques
  - A technique is a goal-directed set of strategies and operations, such as modeling, simulation, and machine learning

Introductory Course (1/3)

- Concepts/techniques students should have learned at the end of one semester
  - converting patterns of data to information (to knowledge)
    - methods for exploring an interesting domain to understand what constitutes 'data' in that domain, and then extract and represent that data
    - methods for analyzing the data to determine what the fundamental issues are for modeling, simulation, and validation
    - methods for deriving and validating information from data, including simple statistical and visualization tools

Introductory Course (2/3)

- representing relationships as models and programs
  - systematic approach to designing, writing, and debugging several hundred line programs, including an understanding of
    - reasons why programming is a way to manipulate patterns as well as a tool for problem solving and modeling, and
    - how it compares with – and augments – other strategies such as the scientific method, mathematics, and classic humanities analytic strategies.
  - moving from ambiguous problem statement to computational formulation
    - method for decomposing and solving the problem as a set or hierarchy of subproblems that solve/implement them.
  - meaning and use of algorithms
    - including the importance of and strategies for scaling

Introductory Course (3/3)

- exploring and validating hypotheses and models
  - methods for using simulations shed light on problems
    - that are ambiguous
    - that don't have an obvious (closed form) solution
  - validation strategies
    - that analyze the results of a simulation against the initial hypotheses and data

2. Core/Foundation for All CS

- Problem
  - what cognitive skills, concepts, techniques, and content should the core consist of
  - how should this be decided
  - how should it be embodied in specific courses and programs
  - what curricular change mechanisms can evolve as the inevitable changes occur
- Approach
  - identify enduring concepts, techniques, and skills to form the foundation for a lean core
  - emphasize mastery and skills across the curriculum
  - develop institutional relationships that build on the

3. Specialization:

- Problem
  - accommodate changing domain-specific interests, both current and as yet unknown future trends
  - develop the increased depth that comes from focused specialization over time
- Approach
  - use the lean core skills and concepts in domain-specific courses and sequences
  - experiment with a variety of mechanisms ranging from one-time experimental courses to major sets of interwoven tracks such as Georgia Tech's Threads
  - work with other departments and institutions to optimize resource use

4. Specialization:

- Problem
  - how to develop deep collaboration, an integrated mindset, among two or more different fields such as computer science and biology
- Approach
  - identify and address some of the barriers to fully-integrated joint majors, and provide detailed example prototypes as models
  - explore joint special courses, e.g., seminars, as testing grounds for more extensive collaborative development
  - encourage cross-departmental student groups

5. Design Under Constraints and

- Problem
  - course concepts and techniques are siloed so students don't make connections between courses
  - wide gap between introduction to concepts and their application to real-world problems
- Approach
  - create projects that tap into student enthusiasms
  - build real artifacts whose performance and effectiveness can be measured
  - treat design as an iterative process throughout the curriculum
  - practice debugging and dealing with real-world issues
  - create integrative capstone-like experiences in all 4

6. Attracting, Selecting, and

- Problem
  - how to attract, select, and prepare students for computationally-oriented research careers
- Approach
  - embed seductive research examples in the introductory courses
  - establish an apprenticeship culture for undergraduates as well as graduate students
  - identify and teach skills needed for success in graduate school
    - analysis of 'related work' and synthesis with research problem
    - identification of hidden assumptions in self and others
    - balancing vision, detail-orientation, rigor, and persistence under failure

Tools for Using this Report

- Draft location
  - http://www.cs.brown.edu/~avd/CRA-E-Snowbird2010-draft.pdf
- Tools
  - recommendations
    - each recommendation section contains background issues, examples, and solution approaches; resulting recommendations are summarized at the end of the section.
    - the complete recommendation set is provided as a single document in Appendix A to give a sense of the set as a whole.
  - references (Appendix B) contain both a bibliography and the complete set of all the URLs that are mentioned in the report.
  - index supports both search and browsing modes

Summary

- Benefits for all undergraduates, not just future computationally-oriented researchers
  - all CS students independent of their career goals
  - non-CS students who just want a particular sequence
- Graduate students benefit from serving as mentors and models for research-oriented undergraduates
- Early identification and nurturing of potential researchers deepens the skills needed for success in graduate school
- ...

Next Steps

- Release final CRA-E white paper
  - posted on CRA website http://www.cra.org/uploads/documents/resources/rissues/CRA-E-Researcher-Education.pdf
  - 'blurbs' in Computing Research News, CACM, ...
- Fork CRA-E "Mark 2"
  - committee chaired by Rich DeMillo
  - website

Comments?
Questions?

http://www.cs.brown.edu/~avd/CRA-E-Snowbird2010-draft.pdf

Example Introductory Courses

- Brown – multiple versions and styles
  - CS15/16, CS17/18, CS19, CS40, CS53, CS931
- CMU – CS15-105 – Principles of computation
- Georgia Tech – CS1315/CS1316 – digital media
- Harvey Mudd – CS for Scientists
- MIT – 6.00 – Introduction to CS and Programming for students with no programming experience
- MIT – 6.01/6.02 – Mobile robots focus on computational engineering
- Princeton – CS116 – The Computational Universe
- Purdue – SECANT: Science Education in Computational Thinking

Cognitive Skills – A Sample Set (1/4)

- Abstractions – creating and validating
- Algorithmic thinking – representing information, working with constraints and automating the process
- Analysis – examining the components and structure of concepts, data, and research results
- Approximations – estimating from data observations and representing in algorithmic form
- Assumptions – identifying and validating
- Automation – representing processes in terms of repeated operations such as iteration and recursion
- Comparing and contrasting – identifying the way in which two or more things are similar and different. The basis for creating abstractions.

Cognitive Skills – A Sample Set (2/4)

- Critical reading and writing – close attention to the semantics of terms, unstated assumptions, and relationships with other work. Related work sections in research papers and reporting on papers in seminars provide training in this skill
- Debugging – detecting pattern anomalies, using isolation strategies
- Decomposing – complex entities into simpler ones
- Designing – integrating user, performance, simplicity, and reliability concerns
- Evaluating results in terms of assumptions and goals
- Exploring – observing and identifying patterns for possible classification
- Hypotheses – pattern recognition and assumption use

Cognitive Skills – A Sample Set (3/4)

- Integrating disparate data and concepts
- Interaction – identifying and representing different roles and their interrelationships; developing communication mechanisms among the different roles
- Logical analysis of representation relationships
- Parallel thinking – identifying sub-components that don't share dependencies
- Patterns – recognition and classification
- Planning – setting goals, developing strategies, and outlining tasks and schedules to accomplish the goal
- Problem solving – working with time and space constraints, decomposing complex problems

Cognitive Skills – A Sample Set (4/4)

- **Reasoning under uncertainty** – reasoning and making decisions based on incomplete and/or uncertain data and models
- **Representing abstractions and their relationships** through notations and language
- **Scaling** – understanding time/space/and power constraints
- **Searching** – focused exploration
- **Symbols and notations** – representing and manipulating information and relationships
- **Synthesis** – combining components of concepts, data, or research into a new construction
- **Tinkering** – manipulating portions of existing entities

Lean Core Examples

- CMU CS Department new curriculum
  - http://www.csd.cs.cmu.edu/education/bscs/currreq.html
- MIT EECS new curriculum
  - http://www.eecs.mit.edu/ug/newcurriculum/index.html
- Stanford CS Department new curriculum
  - http://csmajor.stanford.edu/Considering.shtml

Lean Core Concepts – A Starter

- Algorithmic thinking and problem analysis
  - problem decomposition
    - divide and conquer
    - levels of abstractions
  - Reasoning
    - correctness, logics, invariants, verification, debugging
    - reasoning under uncertainty, probability
    - planning, learning…

Lean Core Concepts – A Starter

- Abstractions (levels of)
  - Identify what to model
    - salients, constraints, pitfalls in assumptions and in approximations
  - How to model it
    - what type
    - multi-disciplinary models
  - How to implement the model
    - solve analytically
    - simulate
      - kinds of simulation
    - visualize the results

Lean Core Concepts – A Starter

- Representation, approximation, and dealing with errors
  - Data
    - types of data to be represented
    - representation techniques and formats, and their limitations
  - Processing techniques and their limitations
    - linearization
    - kinds of simulation
    - granularity in spatio-temporal sampling

Lean Core Concepts – A Starter

- Constraints on computation and computational complexity
  - Models of computations
    - automata and grammars
    - computation graphs
    - dataflow and Petri Nets
    - ATN
  - Constraints and tradeoffs in time, space, power, ...
    - fault-tolerance, reliability
  - Complexity, intractability, undecidability

Lean Core Concepts – A Starter

- Data structures and algorithms
  - (the usual and growing collections)
  - Graphs and networks
    - physical
    - virtual
    - social
    - hypertext

Lean Core Concepts – A Starter

- Transformation and Patterns
  - Transformation
    - mapping between representations
    - rule-based systems
  - Patterns
    - defining->searching v.s. discovering/recognizing
    - machine learning
    - planning
  - Language models

Lean Core Concepts – A Starter

- Information, Knowledge, and Machine Learning
  - Information
    - data models
    - query languages
    - data integrity
  - Knowledge
    - representaton
    - logical reasoning and cognition
    - natural language processing
  - Machine learning
    - supervised and unsupervised learning
    - robotics
    - data mining

Lean Core Concepts – A Starter

- Communication and coordination
  - Abstraction layers and protocols
  - Models such as
    - synchronous/asynchronous
    - broadcast/P2P
    - client-server
    - shared memory/message-passing
    - blackboard architecture
    - cloud
  - Error handling
    - concurrency control problems and deadlock

Lean Core Concepts – A Starter

- Flow of control
  - Sequential
  - Conditional
  - Iteration
  - Recursion
  - Parallelism
    - co-routines
    - threads and processes
    - multi-processing
    - multi-core
    - distributed
  - Non-deterministic computation

Lean Core Concepts – A Starter

- Optimization
- The human element
  - Why the human element matters
  - Perception
  - Cognition
  - Interaction
  - Social dynamics

Techniques – Examples (1/2)

- Abstraction mechanisms
- Combinatorics
- Distributed processing
- Exploration of data-intensive subjects
- Machine learning
- Modeling
- Numerical Methods

Techniques – Examples (2/2)

- Programming
- Proof techniques
- Scientific method
- Simulation
- Symbol manipulation
- System design

NRC – A Framework for Science Education

A Framework for Science Education

Preliminary Public Draft

Committee on Conceptual Framework for New Science Education Standards

Board on Science Education
Division of Behavioral and Social Sciences and Education

NATIONAL RESEARCH COUNCIL
OF THE NATIONAL ACADEMIES

A FRAMEWORK FOR SCIENCE EDUCATION

PRELIMINARY PUBLIC DRAFT

This document is an interim draft of a report from a committee of the National Research Council (NRC) on K-12 science education in U.S. schools. It is being made public so that the authoring committee can receive comments and suggestions from interested practitioners, researchers, and the public to inform its final product.

The majority of the document we are releasing now for public comment consists of articulation of the three dimensions of the committee's framework. Each of the three dimensions of the framework–disciplinary content, cross-cutting elements, and science practices—is described in a separate section following this introduction.

Please note that full citations are not included in this version. They will be included in the final report when it is released.

NRC – A Framework for Science Education

Contents

COMMITTEE ON CONCEPTUAL FRAMEWORK FOR NEW SCIENCE EDUCATION STANDARDS

HELEN R. QUINN *(Chair)*, SLAC National Accelerator Laboratory, Stanford University
WYATT W. ANDERSON, Department of Genetics, University of Georgia, Athens, GA
TANYA ATWATER, Department of Earth Science, University of California, Santa Barbara
PHILIP BELL, Cognitive Studies in Education, University of Washington, Seattle
THOMAS B. CORCORAN, Teachers College, Columbia University
RODOLFO DIRZO, Department of Biology, Stanford University, Stanford
PHILLIP A. GRIFFITHS, Institute for Advanced Study
DUDLEY R. HERSCHBACH, Department of Chemistry & Chemical Biology, Harvard University
LINDA P.B. KATEHI, University of California, Davis
JOHN C. MATHER, NASA Goddard Space Flight Center
BRETT D. MOULDING, Utah Partnership for Effective Science Teaching and Learning,
JONATHAN OSBORNE, School of Education, Stanford University, Stanford
JAMES W. PELLEGRINO, School of Education & Social Policy, University of Illinois,
STEPHEN L. PRUITT, Office of the State Superintendent of Schools, Georgia Department of Education
BRIAN REISER, School of Education & Social Policy, Northwestern University
REBECCA R. RICHARDS-KORTUM, Department of Bioengineering, Rice University
WALTER G. SECADA, School of Education, University of Miami
DEBORAH C. SMITH, Department of Curriculum & Instruction, Pennsylvania State University

HEIDI A. SCHWEINGRUBER, *Deputy Director*
THOMAS KELLER, *Senior Program Officer*
MICHAEL A. FEDER, *Senior Program Officer*
MARTIN STORKSDIECK, *Director*
KELLY A. DUNCAN, *Senior Program Assistant*