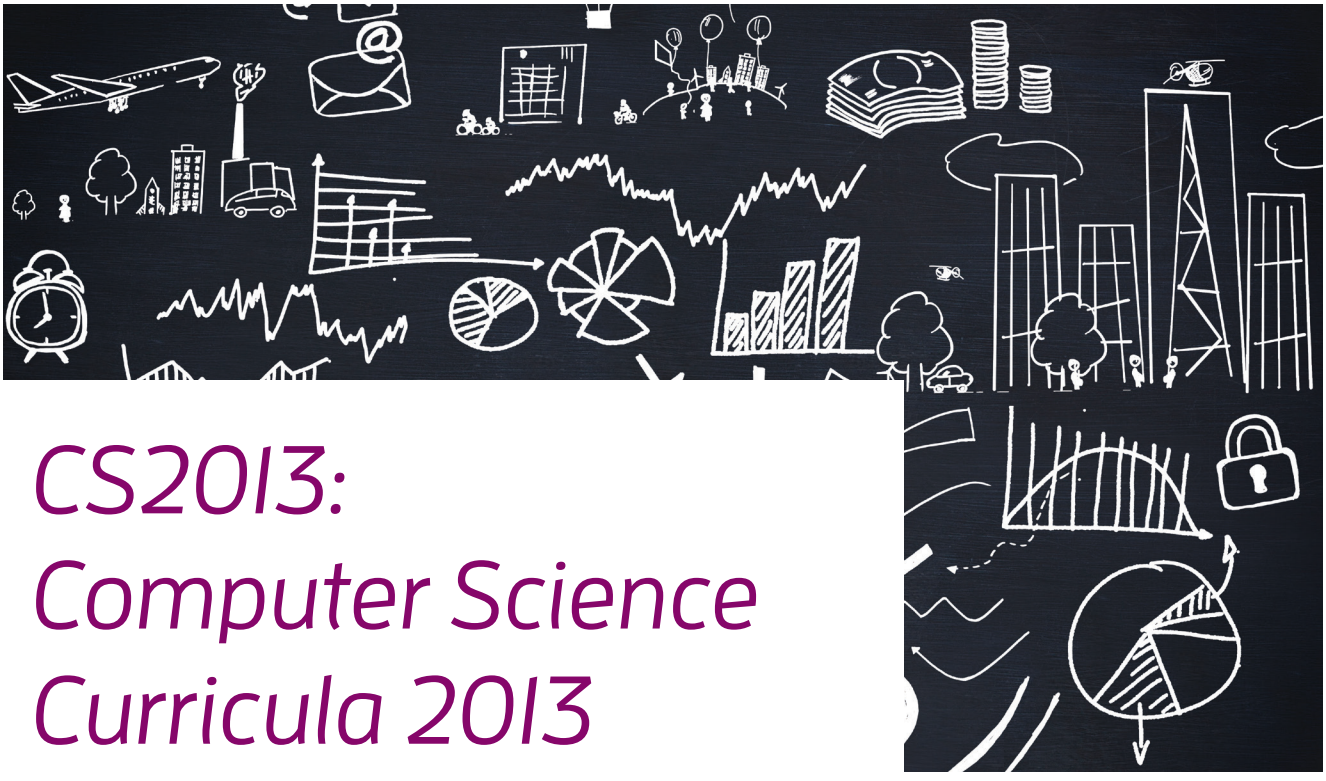

CS2013: Computer Science Curricula 2013

Steven Roach, Exelis

Mehran Sahami, Stanford University

An overview of major changes in ACM/IEEE CS's Computer Science Curricula 2013—the most recent update of the undergraduate education guidelines for computer science.

Just over a year ago, the Association for Computing Machinery and the IEEE Computer Society jointly published the *Computer Science Curricula 2013* (*CS2013*) (www.acm.org/education/CS2013-final-report.pdf), an update to the undergraduate computing programs guidelines they've been publishing about once every decade since 1968.

If you're like most people, undergraduate computer science curricula guidelines aren't foremost on your mind, unless you are or soon will be an undergraduate student, a faculty member teaching undergraduates, or someone concerned with computer science accreditation or the funding of computer science programs and research. *CS2013* provides a baseline for expectations of computer science graduates. Thus, you might also care about this if you're hiring and managing or depending on technology created by new CS graduates.

Computing has changed since 2001, which was the last time the guidelines were completely updated (though there was a partial update in 2008). We started *CS2013* by asking how the previous guidelines were used; we surveyed approximately 1,500 computer science (and related disciplines) department chairs and undergraduate studies directors in the US, and an additional 2,000 department chairs internationally. Respondents were from a wide range of institutions—research universities, teaching universities, undergraduate-only and liberal arts institutions, and community colleges. These institutions range in size from less than 1,000 to more than 50,000 students.

According to the survey, the Body of Knowledge (BOK) is the most-used feature of the guidelines. In addition, respondents' two most common comments about the BOK were that new topics—such as security, distributed

0018-9162/15/$31.00 © 2015 IEEE

EDITOR **ANN E.K. SOBEL**
Miami University; sobelae@muohio.edu

and parallel processing, mobile computing, networking, and professional skills—need to be added, and that parts of the BOK from 2001 and 2008 are indeed still relevant. We view both of these as positive signs of a maturing but still vibrant and dynamic field.

NEW TOPICS

The BOK is divided into knowledge areas (KAs) that embody a set of related topics. New KAs have been added to address major changes in the field in the past 14 years.

Recent headlines are proof enough that the world increasingly relies on information technology, and as a result we're more vulnerable to attacks on our information systems. The new Information Assurance and Security KA encompasses the set of technical and policy controls and processes intended to protect and defend information systems by ensuring their confidentiality, integrity, and availability while providing for their authentication and nonrepudiation.

To addresses issues related to the design and development of software applications that reside on Web-, mobile-, industrial-, and game-specific platforms, we developed the Platform-Based Development KA. Such platforms are often characterized by specialized API use, distinct delivery/update mechanisms, and being abstracted away from the machine level. Although a number of platforms have become prominent, we didn't recommend specific platforms for every CS program. Instead, many popular platforms are highlighted.

The growth in multiprocessor computing, multicore processors, and distributed datacenters continues—indeed, it's now difficult to buy a single-processor machine. Although *CS2008* identified this trend, *CS2013* directly addresses it by changing it from a largely elective topic to a core component of undergraduate computing curricula. The Parallel and Distributed Computing KA addresses the logically simultaneous execution of multiple processes, in which operations have the potential to interleave in complex ways.

Systems KA has increased emphasis on machine learning and data mining. And the Social Issues and Professional Practice KA reflects the past decade's shift in understanding intellectual property in the digital domain and digital rights management, the need for awareness of global issues such as software piracy and how computing's rapid changes impact society, and a growing concern for privacy.

> CS programs need new approaches to cover essential material in the available time.

FUNDAMENTAL IDEAS

In her keynote lecture at the IEEE International Conference on Software Engineering Education and Training (CSEE&T) 2011, Mary Shaw argued that we need to teach "fundamental ideas in the context of current practice" (http://conferences.computer.org/cseet/2011/CSEET_2011/downloads/CSEET_2011_Keynote_MaryShaw.ppt). Her list of fundamental ideas included abstraction, linear versus exponential growth, problem-solving approaches, tradeoffs, symbolic representations, logic, models, and correctness. The KAs in *CS2013* include such fundamental and enduring concepts as differences between best, expected, and worst-case behaviors of an algorithm; finite state machines; data structures; memory hierarchies; caching; and parallel versus sequential computation.

The current report also features extensive revisions of existing KAs from previous curricular volumes. The Architecture and Organization KA has increased emphasis on multicore parallelism, virtual machine support, and power constraints. The Computational Science KA includes elective material to prepare students for cross-disciplinary work such as computational biology, bioinformatics, and eco-informatics. The Intelligent

Depending on your point of view, *CS2013* might pay too much or not enough attention to systems. The term *computer systems* spans operating systems, parallel and distributed systems, communications networks, and computer architecture. Although these are often taught as independent courses, they share fundamental concepts within their respective cores, including computational paradigms, parallelism, state and state transitions, cross-layer communications, and scheduling. A new KA, Systems Fundamentals, presents a view of systems concepts common across existing KAs. We hope this encourages new approaches to covering these topics.

Despite growth in undergraduate software engineering programs and the creation of independent software engineering curricular guidelines, many computer science graduates still go into careers in software development. Part of our approach to helping CS programs address this need is the KA on Software Development Fundamentals (SDF). Regardless of the platform and language used, fundamental concepts in software creation extend beyond coding simple programs. SDF

extends the Programming Fundamentals KA from *CS2001* by drawing basic algorithm analysis material from the Algorithms and Complexity KA, development processes from the Software Engineering KA, fundamental data structures from the Discrete Structures KA, and programming language concepts from the Programming Languages (PL) KA. Material specific to particular programming paradigms (such as object-oriented or functional methods) was moved to PL for a more uniform treatment with complementary material. From a curriculum design standpoint, this separates fundamental concepts (for example, iteration and conditional execution) from the implementation specifics related to a particular language. We believe this opens the door to selecting languages best suited to the student populations at individual institutions while encouraging coverage of professional software development practices early in the curriculum. The Software Engineering KA is still prominent in the guidelines, and it has been updated to include modern software development practices.

FLEXIBILITY IN CURRICULUM DESIGN

Although the computer science field is expanding, the number of hours students spend in undergraduate programs isn't. CS programs will need to consider new approaches to covering essential material in the available time. Curriculum design is, in part, a resource allocation problem. *CS2013* manages the amount of material to be presented to undergraduates in two ways. First, some previous material has reduced emphasis; second, by specifying essential, highly desirable elective topics and providing better guidance as to the level of mastery expected for each topic, instructors have more flexibility in choosing the material to present.

Previous guidelines labeled topics as either core or elective, implying that every core topic is required. However, even some strong CS programs were missing at least one hour of core material. *CS2013* specifies Tier 1 and Tier 2 core topics. Tier 1 topics should be required in every CS curriculum. Tier 2 topics are highly desirable, and programs are expected to cover the vast majority of them. However, programs may sacrifice some Tier 2 topics to provide students with greater depth in other areas. A CS curriculum should cover 90 to 100 percent of the Tier 2 topics, with 80 percent considered a minimum.

CS2013 includes examples of fielded courses to illustrate how topics can be covered and combined in diverse ways.

A common problem with BOKs is that they often identify topics without giving guidance on the required level of mastery. *CS2013* uses familiarity, usage, and assessment as the levels of mastery for each topic. *Familiarity* implies that the student has basic awareness and understanding of a concept. *Usage* implies that the student can use or apply a concept in a concrete way, for example, including it in a program or proof. *Assessment* implies that the student can consider a concept from multiple viewpoints or justify the selection of a particular approach to solve a problem. Each topic is matched with one or more learning outcomes to make the BOK curricular expectations clearer.

CS2013 includes examples of actual fielded courses—from a variety of universities and colleges—to illustrate how topics in the KAs can be covered and combined in diverse ways. It also contains examples of the different ways a larger collection of courses can be put together to form a complete curriculum. Providing these examples promotes greater cross-pollination of educational ideas within the computing community as well as ongoing engagement through encouraging educators to share new courses and curricula from their own institutions (or others that they are familiar with) with the rest of the community.

Any CS curriculum should prepare graduates to succeed in a rapidly changing field; thus, it must prepare students for lifelong learning and include professional practice elements—communication skills, working in teams, ethics, and so on—as components of the undergraduate experience. CS2013 provides guidelines that, when implemented, will enable students to integrate theory with practice, to recognize the importance of abstraction, and to appreciate the value of good engineering design. CS2013 is not a minimal standard against which a program can be evaluated, but a guideline for fostering excellence in CS undergraduate education.

ACKNOWLEDGMENTS

The authors acknowledge the 17 members of the Computer Science Curricula steering committee as well as the hundreds of contributors and reviewers who contributed to the CS2013 report.

STEVEN ROACH is an associate principle software engineer in information systems at Exelis Inc. Contact him at roach.steven1965@gmail.com.

MEHRAN SAHAMI is a professor and the associate chair for education in the Department of Computer Science at Stanford University. Contact him at sahami@cs.stanford.edu.