Experiences Mapping and Revising Curricula with CS2013

David Reed (Moderator)  
Creighton University  
Department of Journalism, Media & Computing  
davereed@creighton.edu

Andrea Danylyuk  
Williams College  
Department of Computer Science  
andrea@cs.williams.edu

Elizabeth K. Hawthorne  
Union County College  
Department of Computer Science  
hawtheone@ucc.edu

Mehran Sahami  
Stanford University  
Computer Science Department  
sahami@cs.stanford.edu

Henry M. Walker  
Grinnell College  
Department of Computer Science  
walker@cs.grinnell.edu

Categories and Subject Descriptors  
K.3.2 [Computers and Education]: Computer Science Education; Curriculum.

General Terms  
Documentation, Design, Standardization.

Keywords  

1. SUMMARY

Roughly once per decade, the ACM and IEEE-Computer Society form a joint task force to survey the discipline and produce curricular guidelines for undergraduate computer science programs. The latest guidelines document, Computer Science Curricula 2013 (CS2013), was released in the Fall of 2013 after multiple rounds of public review and feedback. A significant novel feature of CS2013 is the structuring of the Body of Knowledge core topics into two tiers. While Core-Tier1 topics are expected to be included in every undergraduate computer science program, individual programs may choose to cover only a subset of Core-Tier2 topics, depending upon institutional size, resources, and goals. CS2013 recommends at least 80% coverage of Core-Tier2, which provides flexibility for designing curricula that meet institutional constraints while still maintaining disciplinary standards.

In addition, CS2013 emphasizes institutional flexibility in the way that topics are packaged into courses. Topics in the Body of Knowledge are organized by Knowledge Areas (such as Software Development Fundamentals, Parallel and Distributed Computing, and Social Issues and Professional Practice), not by courses. Curricular designers are encouraged to package topics into courses in creative and innovative ways, and subsequently sequence those courses to form complete curricula. To support the design of innovative curricula, CS2013 includes an appendix with more than 80 course exemplars: syllabi of existing courses with mappings to Knowledge Area topics and learning outcomes. The appendix also includes four complete curricular exemplars from a variety of schools (a research university, two liberal arts colleges, and a community college). These curricular exemplars show four alternative ways that topics from the CS2013 Body of Knowledge can be packaged and sequenced to construct programs of study. More than just documenting current practices, these exemplars are also forward-looking, describing proposed revisions that would bring current models more in line with CS2013 recommendations.

This panel brings together the four authors (or co-authors) of the CS2013 curricular exemplars to discuss the process by which they developed these exemplars and how CS2013 recommendations are driving changes to their existing curricula. The authors will share insights and also provide guidance for educators who are considering reviewing their own curricula in light of CS2013.

2. ANDREA DANYLYUK

Mapping the Williams College CS curriculum against the CS2013 core was an extremely interesting and helpful exercise at a time when the department was holding extensive curricular discussions in preparation for an external review. In order to ensure that students are free to explore a wide range of academic subjects in the liberal arts tradition, Williams places a limit on the number of courses that may be required for a major -- typically 9. While students are free to take many more courses in their majors, our challenge is to provide a curriculum that gives our students good breadth and depth, regardless of the trajectory they follow.

For our analysis, we mapped our required introductory and core sequence, as well as what a "typical" CS major might take. Specifically, we assumed the student would take the 6 intro and core courses, Discrete Math and three CS electives. (When computed over the last three graduating classes, we see that students take, on average, 3.6 electives.) The specific electives we used for the mapping were Digital Design and Modern Architecture, Artificial Intelligence, and Operating Systems.

We found that our common core provides fairly good coverage of the CS2013 core, and electives bring our curriculum even closer in line with the CS2013 recommendations. Doing the mapping reaffirmed our belief that we are especially strong in software
development fundamentals, theory, and programming languages. While our electives provide very strong coverage of systems topics, it is possible for students to choose elective paths that steer clear of those. We were pleased to find that our curriculum provides better coverage of "new" areas (information assurance, parallelism and concurrency) than we predicted. It also confirmed that we are less good in formal software engineering processes. We are still in discussions about the shape of our major, but we have discussed several options that would give students additional formal exposure to software engineering and that would more explicitly steer them to a set of electives that together would ensure broad coverage of CS.

3. ELIZABETH K. HAWTHORNE

Mapping the CS2013 core curriculum against two different types of associate degrees was done collaboratively with Melanie Williamson and Cindy Tucker of Bluegrass Community and Technical College in Lexington, KY (www.bluegrass.kctcs.edu). According to the American Association of Community Colleges, the Associate in Science (A.S.) degree prepares student to transfer to upper division baccalaureate-degree programs, with half of the coursework in general education requirements. Bluegrass’ A.S. degree mapped four core courses against CS2013: (1) Introduction to Computer Programming, (2) Introduction to Program Design, Abstraction and Problem Solving, (3) Introduction to Software Engineering and (4) Discrete Mathematics. The more discipline-focused Associate in Applied Science (A.A.S.) degree can similarly facilitate transfer or lead directly to employment, with only one-third of the requirements in general education. Bluegrass’ A.A.S. degree mapped seven courses to the core of CS2013: (1) Computational Thinking, (2) Introduction to Computers, (3) Computer Hardware and Software, (4) Internet Technologies, (5) Introduction to Networking Concepts, (6) Database Design Fundamentals, and (7) Security Fundamentals.

While performing this detailed mapping exercise, Bluegrass discovered the A.S. degree covers the CS2013 core very well in the knowledge areas one would expect at a freshman-software level and completely satisfies all of the general education components required by Kentucky’s four-year universities. The mapping exercise also confirmed the A.A.S degree, while it does not meet all general education requirements at the four-year level, covers the CS2013 core to a greater degree than the A.S. degree due to the additional seven core Computer and Information Technologies classes taken. Bluegrass was extremely pleased with the mapping exercise and continues to explore additional transfer options for their students.

4. MEHRAN SAHAMI

Mapping the CS curriculum at Stanford against the CS2013 guidelines led to insights not only with respect to CS2013, but also in regard to broader issues in our department's undergraduate program. Stanford has a track-based CS curriculum which requires all students to complete: (i) six "core" classes, (ii) four to five classes in one of nine depth areas (tracks) of a student's choosing, and (iii) two to four additional CS elective courses. While the multitude of options make it impossible to map all paths through our curriculum, solid insight was still gained by examining a single typical program that includes the six core courses, a standard path through the Systems track, and the minimal required selection of popular CS elective classes. This common program provided good coverage of the Core-Tier1 and Tier2 topics in CS2013. However, the mapping process also made us realize that other paths through the major might provide fairly different levels of Core coverage. For example, the coverage of many Core-Tier2 topics (e.g., in Architecture and Organization and Intelligent Systems) is quite dependent on students’ choice of track and elective courses. This led us to the decision that trying to prevent outlier cases (students who craft very uncommon programs that still somehow manage to fit within our program requirements) was a lower priority than optimizing the common cases of what most students would be learning in our program.

At a more detailed level, the mapping helped to identify topical redundancies between courses and also rationalize coverage of some topics (and removal of others) as we considered course updates. We also discovered differences (sometimes significant) between the topical coverage provided by different instructors of the same course. We found that it was often possible to cover CS2013 learning outcomes not already in our curriculum by making modest modifications within existing courses. We also recognized that our quarter-based academic calendar affords a larger number of required courses in our curriculum and thus makes it easier to cover a broader range of topics from CS2013.

5. HENRY M. WALKER

Following a long-standing practice, Grinnell's CS faculty undertook a multi-pass approach to this curricular review. In the first pass, course syllabi from our multi-paradigm, three-course introductory sequence were examined, and hours estimated for each topic. Challenges in counting arose when one class session touched several CS2013 Knowledge Units (e.g., code development for an insertion sort might include loops, conditionals, arrays, functions and parameters, algorithm development, and sorting). After several passes, the allocation of time was judged an honest representation of hours spent, while also capturing appropriate themes and emphases in each course.

During and after a 3-day workshop in summer 2013, Grinnell's CS faculty next divided courses into groups (e.g., languages, systems, software development, etc.), and pairs of CS faculty collaborated to review each course in a group according to the both learning outcomes and level of mastery. After assembling results in a combined spreadsheet, we were surprised at how little of the Core-Tier1 topics was covered in our curriculum. We took it as a challenge to consider how to combine the new perspectives in CS2013 with our own focus on deep student understanding of algorithms, theory, and programming languages, a focus which our students and alumni tell us has long-standing value in their careers. Since a small liberal arts college like Grinnell limits a major to 32 credits (typically 8 4-credit courses), meeting CS2013 recommendations required substantial creativity for both basic requirements and general advising. Once we had curricular mappings for learning outcomes, we used a formula to convert "learning-outcome hours" to in-class hours. With Grinnell's spiral approach to courses, hours spent meeting an objective in one course may not be allocated over several courses. Thus, Grinnell's CS faculty debated how best to report such hours.

Overall, utilization of CS2013 in the review of Grinnell's CS curriculum represented an extensive collaborative enterprise, including faculty members Janet Davis, Samuel A. Rebelsky, John David Stone, Henry M. Walker, and Jerod J. Weinman. Reporting of the final curricular mapping entailed extensive collaboration by Rebelsky and Walker.