1 Introduction

In the United States, students enter college with varied computing backgrounds: generally a good deal of experience as a computer user (via computers, game devices, and phones), but little formal training in computer science. At the end of their degree programs, the successful students are capable of starting careers as computing professionals: having both theoretical and practical knowledge of computer science, they have become computer scientists.

They enter college as computer users, then take courses, do projects and activities, interact with others, have a variety of experiences, and come out ready to join the computing profession. How do they get there?

2 Research question

“How do students develop into computer scientists?” is a broad question, one that encompasses disciplinary knowledge and skill, but also an understanding of the social and professional context. As John Seely Brown relates about physics:

The developmental psychologist Jerome Bruner made a brilliant observation years ago when he said we can teach people about a subject matter like physics–its concepts, conceptual frameworks, its facts–and provide them with explicit knowledge of the field, but being a physicist involves a lot more than getting all the answers right at the end of each chapter. To be a physicist, we must also learn the practices of the field, the tacit knowledge in the community of physicists that has to do with things like what constitutes an “interesting” question, what proof may be “good enough” or even “elegant,” the rich interplay between facts and theory-formation, and so on... Acquiring this expertise requires learning the explicit knowledge of a field, the practices of its community, and the interplay between the two. And learning all this requires immersion in a community of practice, enculturation in its ways of seeing, interpreting, and acting. [2]:

A similar distinction could be made in computer science, between knowing computer science and being a computer scientist.

The above broad question could be examined by looking at student development over a number of different aspects, from when they enter college to when they finish. I suggest these as a start: how do students change over time in terms of

1. Their view of the computing discipline: what it entails, and what its important concepts and skills are;
2. Their broad understanding of the computing discipline: how they understand fundamental computing concepts and their interrelationships;
3. Their views of their education: how they see their roles and those of their peers and their instructors in the educational process;
4. Their career aspirations and expectations.

For each of these we can ask the following questions: what changes are observed, what points within their academic careers do they occur, and what courses and experiences are related to these changes.
3 Results and Impact

Understanding how, when, and why students change has great potential for affecting the way that computer science is taught.

- Understanding how students develop in a computing program in response to formal coursework could support targeted and effective changes in curriculum to improve the learning of computer science. It should suggest changes in emphasis of material, and may provide guidance for reordering topics.

- Understanding how students develop in a computing program in response to their external experiences (such as internships) could help us to better advise our students and more effectively involve industry in our students’ education.

- Understanding why students stay in computer science degree programs would support interventions to increase retention. This study will identify important issues: how students see their own education as they go through their degree program, and how they come to identify with computing as a discipline and themselves as members of the computing community.

- Understanding the factors that lead students to study computer science, and what they expect from a computer science education, should help us to recruit students by targeted means. If we find that these factors vary across different gender and socioeconomic groups, it would be extremely useful in recruiting and retaining a diverse student population.

4 Related work

Perry’s study [6] of Harvard students is the best known longitudinal study of student development at university. Based on interviews with multiple cohorts of students, he developed a model of how students progress in stages from dualism (the truth is out there, and authorities will provide it) to relativism (knowledge is constructed by individuals, based on evidence, an evolving process) as they go through their college years. Among other things, this work demonstrates the value of longitudinal study of individuals, and suggests that interviews can be used effectively as a way of monitoring development.

Bruner’s notion of fundamental ideas [3] is that some central ideas in science can be learned at every level of education to different degrees, and suggests a spiral approach to teaching, where the same ideas are revisited at many levels with greater sophistication. Schwill [9, 10] applied this idea to computer science, defining criteria under which a concept might considered to be fundamental, identifying a number of these concepts, and suggesting that curricula could be organized around these concepts. These suggest that development could be monitored by examining the understanding of fundamental ideas as a student progresses – the understanding of some key ideas could serve as a proxy for the conceptual understanding of computing in general.

The “commonsense computing” work [1, 11] examined student preconceptions: the conceptual knowledge about computing that students have when entering university. This work has examined topics like algorithm analysis and design, logic and probability, debugging, and concurrency, and has found students have a fairly rich understanding of some topics. This work also supports the fundamental ideas concept, as it identifies topics that the students have learned before college instruction. It also suggests data collection and analysis techniques that can evaluate the sophistication of this understanding.

Meyer and Land [5, 4] have developed a theory of learning with “threshold concepts”: concepts whose learning transforms the student’s perspective of the discipline. These transformations often include changes in identity: a progression from outsider to insider in a discipline. As we are interested in students becoming “insiders”, this work suggests that looking for transformational experiences may be valuable. Some of the threshold concepts work has been in computing, see [7, 12] for example.

Schulte and Knobelsdorf’s study [8] of German undergraduates and their attitudes toward computing characterizes students from two different groups – computing majors, and psychology majors – and identifies different perceptions of computing in terms of self image, world view, and computing habits. They show significant differences between students who chose computing and those who did not: for example, the non-computing students viewed “professional” computing as something like tech-support and systems.
administration, while computing students viewed it as primarily design. The transition from user to designer is an opportunity for the computing students, but a barrier for the others. In addition to providing a detailed model of “insiders” and “outsiders”, this work demonstrates the value of using computer biographies as a way of collecting analyzable data.

References


