2010

Computer Science And The Liberal Arts: A Philosophical Examination

H. M. Walker

Charles F. Kelemen
Swarthmore College, cfk@swarthmore.edu

Computer Science and the Liberal Arts: A Philosophical Examination

HENRY M. WALKER
Grinnell College
CHARLES KELEMEN
Swarthmore College

This article explores the philosophy and position of the discipline of computer science within the liberal arts, based upon a discussion of the nature of computer science and a review of the characteristics of the liberal arts. A liberal arts environment provides important opportunities for undergraduate programs, but also presents important constraints. A well-designed program can flourish in this environment, and evidence indicates that a liberal arts program in computer science can indeed succeed well.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*; *curriculum*

General Terms: Theory

Additional Key Words and Phrases: Liberal arts, undergraduate education

> How can computer science fit within the liberal arts? I thought computer science had to do with engineering and real-world applications, and their concerns seem different from those of the liberal arts.

Comments of this type arose frequently in the 1980s and have continued periodically to the present day. Just recently, a prospective student asked a question of this type to one of the coauthors of this article. This article seeks to provide a careful answer.

Some historical comments may provide some initial perspective. ACM published its first comprehensive curricular recommendations in 1968 [ACM Curriculum Committee on Computer Science 1968] and updated those recommendations in 1978 [Austing et al. 1979], 1991 [ACM-IEEE-CS Joint Curriculum Task Force 1991], and 2001 [ACM / IEEE-CS Task Force on the Curriculum 2001]. Although these curricula provided fine insight, they treated all institutions as being similar; the same recommendations were to apply to technical schools, research-oriented universities, and liberal arts colleges.

In the 1970s and early 1980s, some educators suggested that large, technically-

Corresponding Author's Address: Henry M. Walker, Department of Computer Science, Grinnell College, Noyce Science Center, 1116 Eighth Avenue, Grinnell, Iowa 50112.
Collaborating Author's Address: Charles Kelemen, Computer Science Department, Swarthmore College, 500 College Avenue, Swarthmore, PA 19081

oriented schools could cover the full ACM curriculum, but the curriculum had to be scaled back in liberal arts colleges. In this view, computer science in a liberal arts setting was often expected to be a watered-down program.

In response, a group of faculty from liberal arts colleges started meeting in 1985 to consider an appropriate philosophy for an undergraduate degree in computer science. With partial funding from the Sloan Foundation, the group published its first "Model Curriculum for a Liberal Arts Degree in Computer Science" in 1986 [Gibbs and Tucker 1986], and revisions have followed in 1996 [Walker and Schneider 1996] and 2007 [Liberal Arts Computer Science Consortium 2007]. These model curricula build on the premise that computer science fits naturally with the liberal arts; and the liberal arts setting reinforces the fundamental concepts, theory, and practice of an undergraduate program in computer science.

For many, the 1986 Model Curriculum [Gibbs and Tucker 1986] demonstrated that a strong, intellectually valid computer science program could indeed reside within a liberal arts setting; computer science could be legitimate within the liberal arts. However, even with the publication of the Model Curricula [Gibbs and Tucker 1986, Walker and Schneider 1996, Liberal Arts Computer Science Consortium 2007], questions continue on various college campuses, with prospective students, and among employers. This article provides a thorough examination of computer science and its role within the liberal arts.

The article begins with a clarification of what might be meant by "computer science." Over the years, various audiences have viewed the field of computing in different ways, and undergraduate programs have appeared in numerous related areas (e.g., information technology, information science). Section 1 gives several perspectives regarding the specific field of "computer science."

A liberal arts curriculum promotes a broad study of multiple disciplines, develops reasoning and analysis, and invites multiple views of problem solving. Section 2.1 outlines several views regarding the nature of the liberal arts and the corresponding characteristics of undergraduate liberal arts programs.

Since a liberal arts environment brings several important strengths to the study of computer science, an undergraduate program in computer science can build from a solid foundation. However, the liberal arts philosophy of breadth and multiple perspectives also injects some practical limitations. Section 2.2 explores both opportunities and constraints on a computer science major within the liberal arts setting. Section 2.3 then examines the impact of computer science on the liberal arts.

Since undergraduate liberal arts programs have been producing majors for years, it is natural to consider how well graduates succeed in highly technical areas, such as computer science. Section 3 reviews some measures that suggest a liberal arts background does indeed have a fine record of success.

Although this article focuses upon perspectives of liberal arts, the article does not seek to imply that all qualities listed here are the exclusive province of liberal arts institutions. Indeed, various qualities described here may be found in many types of schools. The point of this article is to present a unifying view of the liberal arts and the place of computer science within that context.

1. THE REALM OF COMPUTER SCIENCE

Over the years, the term "computer science" has come to be used in numerous ways by various social and cultural groups. Within an academic setting, a description of the discipline of computer science can clarify the role of computer science in relationship to other areas of science and technology and also can examine the central core of computer science within interdisciplinary projects and endeavors.

1.1 The Nature of Computer Science

Within society as a whole, a general citizen may use the terms "computing" and "computer science" to refer to a wide range of topics. In beginning a discussion of computer science and the liberal arts, therefore, some clarification may be helpful.

(1) Some people use "computing" to refer to the use of multiple paradigms to solve problems, drawing upon reasoning, logic, analysis, hypothesis testing, and formal problem-solving methodologies.
(2) To some, "computing" refers to the computer hardware, software, printers, networking, etc., that comprise an organization's electronic infrastructure.
(3) Some consider "computing" to mean the user directives and low-level skills involved in running specific software packages (e.g., what key strokes a user should type to perform a desired spreadsheet computation).

Although these descriptions may not be completely disjoint, they illustrate different emphases and perspectives. To clarify responsibilities, a department of "computer science" typically focuses on the problem solving of the first perspective, a department of "information technology services" or "ITS" supports and maintains the electronic infrastructure, and a "help desk" or similar organization typically provides tutoring or noncredit workshops for running specific packages.

Similarly, an undergraduate curriculum for computer science emphasizes approaches to problem solving, algorithms and data structures, social and ethical issues of the use of computers, and a consideration of theoretical and practical limits of algorithmic problem solving. Although a computer science curriculum may include hardware, specific languages, and applications to illustrate concepts, computer science focuses upon principles, formal properties, and problem-solving methodologies.

Additional notes on the nature of computer science may be found in discussions of liberal arts curricula, such as [Gibbs and Tucker 1986, Walker and Schneider 1996, Liberal Arts Computer Science Consortium 2007].

1.2 Computer Science in an Interdisciplinary Setting

As computer science has evolved as a discipline, researchers and developers have integrated insights and advances from many related fields. For example, Computing Curricula 1991 [ACM-IEEE-CS Joint Curriculum Task Force 1991] argues that the discipline of computing integrates three fundamental processes:

—*theory:* from mathematics,
—*abstraction:* based upon the scientific method, and
—*design:* from engineering [ACM-IEEE-CS Joint Curriculum Task Force 1991].

In recent years, computer science has connected with many interdisciplinary efforts (e.g., bioinformatics, neuroscience), and specific boundaries for computer science itself may be fuzzy. However, even in wide ranging research, a computer science perspective likely highlights underlying algorithms, data representations, and principles as part of an overall research team.

2. THE REALM OF LIBERAL ARTS

Programs in the liberal arts build upon a centuries-old perspective that places an emphasis on concepts, principles, and a breadth of education. Students and faculty strive to bring multiple viewpoints to bear in addressing topics and issues. By its very nature, computer science fits into this environment in at least two ways.

—The liberal arts shapes the study of computer science, as a discipline that inherently draws upon insights and perspectives from many subjects.

—Computer science itself contributes insights and perspectives to an overall liberal arts environment.

2.1 Characteristics of Liberal Arts Programs

Historically, the notion of liberal arts has evolved over centuries. The focus has been on education of the whole person, and medieval universities in Europe emphasized the Trivium (grammar, rhetoric, and logic) and the Quadrivium (geometry, arithmetic, music, and astronomy). More recently, areas of study have expanded to include the arts, language, philosophy, history or social studies, mathematics, and science. Just as liberal arts History and English majors may not become professional historians or writers, liberal arts computer science majors may not choose to become professional computer scientists. A liberal arts computer science major in the context of other liberal arts courses must provide a strong foundation for life and lifelong learning independent of the career choice of the student. Of course, liberal arts computer science majors must also be prepared deeply enough in computer science so that they can pursue graduate study or entry into computer science work. The 2007 Model Curriculum [Liberal Arts Computer Science Consortium 2007] gives this description:

> Liberal arts programs in computer science generally emphasize multiple perspectives of problem solving (from computer science and other disciplines), theoretical results and their applications, breadth of study, and skills in communication. In addition to the material content of computer science, the algorithmic approach is a very general and powerful method of organizing, synthesizing, and analyzing information. Three general-purpose capabilities that are among those fundamental to a liberal arts education are the ability to organize and synthesize ideas, the ability to reason in a logical manner and solve problems, and the ability to communicate ideas to others. The design, expression, and analysis of algorithms and data structures utilizes and contributes significantly to the development of all three capabilities. [Liberal Arts Computer Science Consortium 2007]

To support breadth and exploration of multiple perspectives, a liberal arts program limits work in one area, and a typical computing curriculum might show the

following balance:

—computer science courses: about 30%,

—mathematics courses: about 10%,

—other science courses: 5 – 10 %,

—nonscience (e.g., humanities, social science) courses: 50 – 55%

In the resulting degree, often labeled "Bachelor of Arts," required computer science and mathematics typically make up about 40% of the overall course work for a degree; about 60% of an undergraduate program is outside the major (outside both computer science and supporting mathematics). In contrast, for degrees typically labeled "Bachelor of Science" or "Bachelor of Engineering," the percentages are reversed, with about 60% of the course work for these degrees being in the major and about 40% being outside.

Considering this breakdown in another way, in a school in which most courses carry four credits, graduation typically requires 31 or 32 courses overall, of which 8–12 will be computer science and supporting mathematics for a Bachelor of Arts degree in computer science. At Swarthmore College, for example, of the 32 courses required for graduation, at least 20 must be outside the major. Similarly, Grinnell College requires 124 credits (the equivalent of 31 4-credit courses) for graduation; a major requires 32 credits (8 courses); and no more than 48 credits (12 courses) within a department can count against the 124 total.

A well-constructed Bachelor of Arts program for computer science can draw upon breadth in several ways. Here are a few examples.

—Many courses outside a major will reinforce work on communication skills (e.g., writing and oral presentation).

—Students will have significant exposure to areas in the humanities and social sciences.

—Liberal arts settings emphasize connections between subjects and interdisciplinary work between departments and programs. For example, computer science may work with a philosophy department on courses that consider ethics within a technological society. Similarly, courses on digital art or electronic music may enrich student's understanding of how computing might fit within other disciplines. Altogether, courses may provide both technical and nontechnical perspectives on common problems.

2.2 Computer Science within the Liberal Arts: Opportunities and Constraints Shape Curricula and Courses

As mentioned earlier, a liberal arts program in computer science requires 8–12 courses. Thus, it must focus on principles, fundamental ideas, underlying concepts, and key examples. When the number of courses is limited, offerings and selections must be particularly careful and strategic. Such a focus is consistent with key finding 2 from "How People Learn" [Bransford et al. 2000, p. 16].

> To develop competence in an area of inquiry, students must: (a) have a deep foundation of factual knowledge, (b) understand facts and ideas

> in the context of a conceptual framework, and (c) organize knowledge in ways that facilitate retrieval and application.

This seems particularly appropriate in a field that changes as quickly as computer science. For example, William Wulf, former President of the National Academy of Engineering, reported that a 2000 workshop calculated the "half-life of engineering knowledge" as between 2.5 years and 7.5 years. Wulf concluded, "half of what we are teaching our students in some fields (computer science, by the way, was the field of 2.5 years) is obsolete by the time they [students] graduate." [Wulf 2002, p. 6]

Liberal arts programs also celebrate multiple views of problem solving, considering insights of many disciplines. Expectations for breadth encourage students to take courses outside their areas of specialization, resulting in at least three practical advantages.

—Liberal arts students learn the terminology, concepts, perspectives, and insights of multiple disciplines. Since software systems focus on application domains, the common language for a project comes from the application, not from computer science. Computing professionals, on any development team, must be comfortable with the perspectives of the application area.

—Course work in the liberal arts often involves team-based assignments and projects, providing insights and experience with group-based activities and interactions. Many computing professionals have observed that significant software development projects now take place in teams; computer scientists work with experts in various application areas.

—Much of the liberal arts brings together multiple perspectives and disciplines. Although research arises from pushing current techniques and ideas further, breakthroughs in research often arise when a person connects different ideas in creative ways.

A common perspective for the liberal arts is that an undergraduate degree should provide a foundation. As Wulf observes, "Every other profession treats at least a Masters Degree as the first professional degree. Engineering is the *only* discipline that believes that the baccalaureate is a professional degree. I think the fact that we have not faced up to that causes all kinds of foolishness." [Wulf 2002, p. 6]

A liberal arts undergraduate program has many opportunities to focus upon fundamentals as well as multiple viewpoints and connections; this results in significant advantages. With a liberal arts foundation, graduate work or apprenticeship easily enhance professional specialization.

2.3 The Liberal Arts Gain from Computer Science

Historically, computer science grew out of electrical engineering and mathematics. As computer scientists began to develop systems that would be easy to use for tasks other than calculation, they found it necessary to study areas such as psychology, biology, and linguistics. At the same time, workers in these and other fields discovered that they could use computer systems to model some of their ideas. Thus, computer science has come to share some topics with disciplines other than mathematics and engineering. A few such disciplines are psychology and biology (human-machine

interface, brain theory, artificial intelligence, genomics, proteomics, bioinformatics), philosophy (logic and artificial intelligence), and linguistics (formal languages, natural language understanding). More generally, computer applications are found in almost every academic discipline, and the creation of useful, innovative computer applications in any discipline requires both knowledge of that discipline and knowledge of computer science.

Computer science also provides problem-solving perspectives for addressing problems in many disciplines. Sometimes called "algorithmic thinking" or "computational thinking" [Wing 2006], methodologies within computer science involve an active, creative process for understanding a problem, designing and organizing solutions, and presenting those solutions in a precise and logical fashion. And, once the algorithmic way of thinking has been mastered, it may be applied to questions of all sorts, independently of any desire to obtain a solution from a computer. Algorithmic thinking can be used to counteract the natural human tendency for quick and easy, but sometimes careless and sloppy, thought. This is not to say that the algorithmic mode of thinking is a panacea. Obviously, there are important questions that require other types of thinking. Algorithmic thinking is, nonetheless, a very powerful tool when added to other modes of thought.

Turning more specifically to a liberal arts environment, the mission statement of Grinnell College, typical of many liberal arts colleges, includes the following elements:

> a mission to educate its students "for the different professions and for the honorable discharge of the duties of life." ... The College aims to graduate women and men who can think clearly, who can speak and write persuasively and even eloquently, who can evaluate critically both their own and others' ideas, who can acquire new knowledge, and who are prepared in life and work to use their knowledge and their abilities to serve the common good.

As this statement indicates, a liberal arts education is to be an education for life, not some short-term vocational preparation. As already noted, computer science interacts with many disciplines and thus is vital for citizens of "the different professions." Algorithmic thinking also provides vital insights for addressing many problems throughout society, and computer science in a liberal arts environment can help students hone their skills in problem solving, abstraction, formalization of vague ideas, careful and critical thinking, management of complexity, and clear and concise communication of ideas. These are important, general skills useful in every discipline and useful for life.

Furthermore, computer systems have become a significant factor in contemporary life. Understanding this technology and its implications, therefore, has become vital "for the honorable discharge of the duties of life." Like any powerful technology, computer systems can be used for the benefit of all or, in the hands of the selfish, for the benefit of a few at the expense of many. Although relatively few liberal arts graduates may enter a computing profession, many liberal arts graduates will be making policy decisions and taking leadership roles within a democratic society. All of these people need to understand opportunities and issues related to technology; they need insights to understand implications and to ask appropriate questions.

People well-educated in the liberal arts with some knowledge of computer science are needed to help decide what computers ought to do.

Overall, computer science has a strong, symbiotic relationship with the liberal arts. Computer science draws upon ideas and perspectives from other disciplines; from this standpoint, computer science might be considered the ultimate of liberal arts disciplines. Further, computer science connects with disciplines throughout the liberal arts and impacts many components of general society; from this standpoint, computer science also has much to offer the general education component of any liberal arts program.

3. SOME OUTCOMES FROM LIBERAL ARTS PROGRAMS

Since liberal arts programs have been producing computer science and, more generally, science graduates for many years, it is natural to ask how well these graduates succeed. As already noted, a liberal arts graduate likely has taken somewhat fewer courses in the major than a graduate with a more technical degree, but the liberal arts graduate also likely has considerable breadth. Direct comparison of various types of graduates can be difficult, but quantitative and anecdotal information provide some hints.

—In a recent NSF study of "Baccalaureate Origins of S&E [Science and Engineering] Doctorate Recipients," only 20 of the top 50 schools (ranked by number of S&E doctorates per 100 graduates) were research-oriented universities. Most of the remaining 30 schools are undergraduate, liberal arts colleges [Burrelli et al. 2008]. The authors are not aware of any recent study that gives figures specifically for computer science. In an earlier study by Franklin and Marshall covering the years 1986–1995, the undergraduate origins of doctoral recipients in CS were consistent with the recent NSF study for Science and Engineering. We suspect that this is still the case.

—Both authors have many former students employed in the computer industry. A 2003 alumna of Swarthmore College who did NOT attend graduate school wrote the following in response to a question about how well prepared she was to enter industry.

> While creating a system that tracks error reports to a main server, I'm learning to manage a large-scale project and large amounts of code. The system has an interface that automatically tracks everything that has been reported, worked on, and changed—and everything is accessible from the Web. When it's released, anyone will be able to download the source-code and use the software. It's exciting to be working on something that people will really use.
>
> Swarthmore understood that the broader picture would serve us best in the "real world"; that, by giving us a holistic view of the discipline instead of narrowly focusing on applied software engineering, we would be better equipped to solve a wider array of problems that arise in real-world situations. Now an engineer at Google, I am constantly aware of the educational advantage this background has given me—although a B.S. in software engineering may teach one how to approach specific

> problems, a B.A. in computer science from Swarthmore has given me the tools necessary to approach any problem.

—One of the authors organized about ten industry panels as part of a series of workshops for high school computing teachers. These workshops, funded by the Noyce Foundation and Roy J. Carver Charitable Trust, provided content, background information, and perspectives to help teachers guide their students through schooling to careers in the computing field. During these panels, every industry panelist, without exception, listed "communication skills" and "the ability to work in groups" as the two most important qualifications for success. Although the ordering of these two areas varied by panelist, specific technical skills never made the top two qualifications on the list. This is consistent with the first key point in the 1999 Business-Higher Education Forum report, "Spanning The Chasm: A Blueprint for Action" [Forum 1999]:

> In addition to providing pertinent data, the core curriculum needs to help students develop flexible and cross-functional skill sets, including leadership, teamwork, problem solving, time management, communication and analytical thinking.

—One of the authors of this article observed Members of Technical Staff years ago during a project at Bell Laboratories. In that environment, all technical people were competent. However, the folks in various lead positions were notable for their communication skills and ability to work with clients, customers, and other team members.

Altogether, the experience of breadth and interdisciplinary work of the liberal arts would seem to have a direct payoff in professional careers.

4. CODA

Overall, a liberal arts program emphasizes general knowledge, multiple perspectives, alternative ways of thinking, and connections among disciplines. This philosophy places constraints on specific requirements for a computer science program, but also encourages strategic choices and a focus on long-term and fundamental ideas. Within a computer science program, liberal arts graduates master core ideas, structures, algorithms, and methodologies. These graduates also have considerable experience with writing, oral communication, and ideas from other disciplines. Such a breadth of background provides a strong base for professional careers as well as explorations into new areas and interdisciplinary challenges.

REFERENCES

ACM / IEEE-CS Task Force on the Curriculum. 2001. *Computing Curricula 2001.* ACM and the IEEE Press, New York.

ACM Curriculum Committee on Computer Science. 1968. Curriculum '68: Recommendations for academic programs in computer science. *Communications of the ACM 11,* 3 (Mar.), 151–197.

ACM-IEEE-CS Joint Curriculum Task Force. 1991. *Computing Curricula '91.* Association for Computing Machinery, New York.

Austing, R., Barnes, B., Bonnette, D., Engel, G., and Stokes, G. 1979. Curriculum '78: Recommendations for the undergraduate program in computer science. *Communications of the ACM 22,* 3 (Mar.), 147–166.

Bransford, J. D., Brown, A. L., and Rodney Cocking, e. 2000. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. National Academy Press, Washington, DC.

Burrelli, J., Rapoport, A., and Lehming, R. 2008. Baccalaureate origins of s&e doctorate recipients. Report 08-311, NSF. July.

Forum, B.-H. E. 1999. *Spanning The Chasm: A Blueprint for Action*. Business-Higher Education Forum, Washington, D.C. available at http://www.bhef.com/publications/pubs.asp.

Gibbs, N. and Tucker, A. B. 1986. A model curriculum for a liberal arts degree in computer science. *Communications of the ACM 29,* 3 (Mar.), 202–210.

Liberal Arts Computer Science Consortium. 2007. A 2007 model curriculum for a liberal arts degree in computer science. *Journal on Educational Resources in Computing (JERIC) 7,* 2 (June), article 2.

Walker, H. M. and Schneider, G. M. 1996. A revised model curriculum for a liberal arts degree in computer science. *Communications of the ACM 39,* 12 (Dec.), 85–95.

Wing, J. M. 2006. Computational thinking. *Communications of the ACM 49,* 3 (Mar.), 33–35.

Wulf, W. 2002. The urgency of engineering education reform, excerpts from the litee 2002 distinguished lecture, auburn university, al march 22, 2002.

Received November 2008; February 2009; July 2009; November 2009; accepted November 2009