Chapter
Patterns of Curriculum Design

Douglas Blank & Deepak Kumar
Department of Mathematics & Computer Science, Bryn Mawr College, Bryn Mawr, PA 19010 (USA)
Email: dblank, dkumar@brynmawr.edu

Abstract We present a perspective on the design of a curriculum for a new computer science program at a women s liberal arts college. The design incorporates lessons learned at the college in its successful implementation of other academic programs, incorporation of best practices in curriculum design at other colleges, results from studies performed on various computer science programs, and a significant number of our own ideas. Several observations and design decisions are presented as curriculum design patterns. The goal of making the design patterns explicit is to encourage a discussion on curriculum design that goes beyond the identification of core knowledge areas and courses.

1. INTRODUCTION

In this paper, we present a perspective on the design of a curriculum for a new computer science program at Bryn Mawr College. Founded in 1885, Bryn Mawr College is well known for the excellence of its academic programs. Bryn Mawr combines a distinguished undergraduate college for about 1200 women with two nationally ranked, coeducational graduate schools (Arts and Sciences, and Social Work and Social research) with about 600 students. As a women's college, Bryn Mawr has a longstanding and intrinsic commitment to prepare individuals to succeed in professional fields in which they have been historically underrepresented. In 1999, the

college decided to add computer science to the college's academic programs. We are currently engaged in the expansion and design of the program. The design of the curriculum is being carried out based on several considerations that are discussed in this paper. The design incorporates lessons learned at the college in its successful implementation of other academic programs, incorporation of best practices in curriculum design at other colleges, results from studies performed on various computer science programs, and a significant number of our own ideas.

2. CURRICULUM DESIGN PERSPECTIVES

Margolis & Fisher have reported, based on a 5-year study on gender issues in computer science at Carnegie Mellon University, that female disinterest in computer science is not genetic, nor accidental, nor inherent to the discipline of computer science, but largely due to three factors: early childhood gender socialization (at home); a combination of adolescence, peer relationships, computer game design, and secondary school social pressures; and the fact that female orientation towards (and concerns about) computing are different from the design of most computer science curricula[1].

The last issue is of particular concern to us. Margolis & Fisher claim that universities have historically developed computer science courses with a male bias. Thus, even the introductory courses in computer science are built around 'male preferences' focusing on the very technical aspects from the very beginning. Further, based on interviews of over 100 female college students, they concluded that the female expression of lack of interest in computer science is really based on a lack of confidence.

At Bryn Mawr, we are currently engaged in the design of a new curriculum for computer science. While there exist prescribed and authoritative guidelines for a curriculum in computer science (the Association for Computing Machinery has announced a new basis for computer science curricula [2]), we are going about the design of our curriculum in an extremely independent and deliberative manner. This is partly in resonance with the findings of Margolis and Fisher, and largely based on our own experiences and similar findings at Bryn Mawr and at other universities. In the design of our curriculum, we are taking the challenge of engagement for women in computer science as our primary concern. Several design considerations have gone into the creation of our computer science curriculum: the context of the program within a women's liberal arts college; the context of computer science courses within the program; the requirements for a major; the design of a minor in computer

science; the design of everyday examples and exercises in all courses in computer science.

First and foremost, it is important to establish a place for computer science in a liberal arts college. In the last few years, most computer science programs at large American universities have shifted their divisional alignment from being a program in the arts and sciences to one belonging to the school of engineering. Therefore, the question naturally arises: *If computer science has become a discipline of engineering, why offer it at a liberal arts college?* Without going into a lengthy response to this question, we will simply summarize it here by pointing out that while there are aspects of computer science that lean towards engineering, there are significantly more aspects of computer science that intersect with the broader goals of the liberal arts. In universities where career preparation is the main emphasis, computer science finds itself in the engineering school. In liberal arts colleges, the emphasis of a computer science program is on gathering, evaluating and disseminating knowledge with foundations in the area of logic, mathematics, and the sciences. We believe that by taking a broader perspective on the nature of the discipline (we have proclaimed it to be a core liberal art of the future) and the constant inclusion of the implications of technology and its use in society as well as other academic disciplines will help stir a wider interest in computer science among students.

The view of computer science as a liberal art also leads to a re-examining of course offerings, as well as the content of various courses offered in the computer science program. Additionally, it leads one to re-evaluate the set of courses that, for an individual student, define a major (or a minor) in computer science. It further impacts the design of exercises and examples that are used in individual courses. In the process of designing our new computer science program for Bryn Mawr College, we have deliberated about all these issues at length. Below, we present some pertinent observations and design decisions that have affected our deliberations. While some of the observations arise out of formal studies conducted elsewhere, some are based on our own experiences and experiences shared by the global community of computer science faculty. We are not presenting them as a prescription for the design of all computer science programs, rather to facilitate a discussion on the underlying issues and their implication for curriculum design.

3. CURRICULUM DESIGN PATTERNS

We are calling our collections of observations and decisions curriculum design patterns after the 'design patterns movement' in objected-oriented

software design [3], which is based on the idea of design patterns developed by the architect Christopher Alexander: 'Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.' [4]

3.1 Computing across the curriculum, not!

Our starting point is to defer from the obvious notion of 'computing across the curriculum'. Several disciplines have adopted such an approach: 'writing across the curriculum' or, more recently, 'mathematics across the curriculum'. Given the pervasive nature of computing in these times, it would be an obvious choice to champion computing across the college as a way to engage women in computer science. However, we strongly feel that affecting another department or program's curriculum *is* an imposition and it is unclear whether it will result in more students engaged (or favorably disposed) into further study in computer science. While there have been documented successes of the *writing across the curriculum* and *mathematics across the curriculum* for non-English majors and non-mathematics majors, there is no evidence that such initiatives lead to more engagement in English and Mathematics majors.

For computer science to become a sought after and engaging field of study, we believe that it is our program's responsibility to demonstrate in effective ways how computing has become pervasive in today's society, and hence we have taken a fresh look at the context in which the program sits at a college like Bryn Mawr. From a curriculum design perspective, our response is to concentrate on courses within computer science. There are several courses that are explicitly created for *all* students at the college, and there are also upper-level computer science electives that are also open to non-computer science students. The objective of the design of these courses is to go well beyond the goal of achieving *fluency* to a more intellectual level discourse of ideas and concepts of which fluency may just be one of the *side effects*. This has resulted in several courses that are offered in the computer science program. These are presented in the next three sections.

3.2 Participation in the freshman seminars

All students entering Bryn Mawr College are required to take two freshman-level *college seminar* courses. These courses are designed as pre-disciplinary expositions that encourage critical thinking in addition to developing strong writing skills. In our computer science program, we have

made the commitment to offer at least one course each year in the college seminar program. This has lead to the design and creation of a diverse range of courses at a pre-disciplinary level with a basis in computer science and technology. The topic areas change from year to year. Here are two recent offerings:

Weaving the web: A course that examines the history of the development of the worldwide web, its current use, and its implications on the global society. Eighteen students enrolled constitute the editorial board of a web-based magazine. Students learn the technology underlying web design and *all* written work is in the form of articles for the web magazine. The scope of the magazine is to discuss issues relating to technology and its implications on the Bryn Mawr community [5].

Robots gone berserk--- A look at robots in film: A course that examines portrayals of robots in film and compares it to the state-of-the-art in Artificial Intelligence and Cognitive Science. Described to students as, 'This College Seminar is not a writing course. It is not a film course. It is not a robotics course. It is not a science fiction course. Although, come to think of it, we will write some papers. We will watch some movies. We will study some robotics. And we will read some science fiction. However, this course is really about thinking. In fact, we will spend quite a bit of time thinking about thinking.' [6]

3.3 A *terminal* CS1 course is terminal, for women

Most universities offer two versions of an introductory level course in computer science: One designed for students who wish to major in computer science; one designed for those who do not. Our claim is that most women who otherwise might go on to major in computer science, due to lack of confidence, self-select themselves into the *terminal* non-majors version of the course. This is a subtle observation that most formal studies about gender equity will tend to overlook and hence is unlikely to be uncovered in any study. At Bryn Mawr, we offer only one version of the introductory course. Traditionally, 3/4 of the students in this course are from non-science majors.

3.4 Upper-level electives are interdisciplinary

In order to further the intellectual engagement of all Bryn Mawr students in computer science, several upper-level computer science courses are designed so that they are also accessible to students in other disciplines. The issue of non-majors not meeting a particular course's prerequisites is addressed by expanding the scope of the course by relating to topics and

disciplines outside of computer science, as well as by including team-oriented exercises where each team is comprised of students from diverse disciplines contributing and exchanging ideas and perspectives. Examples include the following courses:

Cognitive Science: Open to students in all disciplines. Also cross-listed in the philosophy department.

Artificial Intelligence: Open to students in all disciplines. Also cross-listed in the Philosophy department.

Digital Multimedia: Open to students of all disciplines (will be offered for the first time in Spring 2003).

Recent Advances in Computer Science: This is a special topics course. Enrollment is open, depending on the specific topic. Example topics: *Computer-related Risks:* Open to students in all disciplines. *Biologically Inspired Computational Models of learning:* Open to all science majors (as well as graduate students).

Thus, by reexamining the notion of *computing across the curriculum* , and beginning from within the computer science program, we have *opened up* access to the field of computer science to the wider community of students at Bryn Mawr College. This has affected the design of courses at the freshman/predisciplinary-level, introductory level, as well several upper-level electives.

3.5 *Humanizing* core computer science courses

Within computer science, the design of each core course is an amalgamation of the concepts underlying the topic of the course (as recognized by the larger computer science community) and the social and cultural implications of the topic in society. For example, consider the course, *Principles of Programming Languages.* This is considered a core course in computer science. It is largely concerned with the principles underlying the design of programming languages. Traditionally flooded with technical content, the course has been redesigned to include case studies of practitioner's lives. There are several biographies and essays available that describe the lives of computer scientists who could be classified as 'programming language experts' [7, 8, 9, 10]. Some of them have been designers of prominent programming languages, while some have led commercial ventures based on programming language products. Thus, a *human element* is included in the course without necessarily sacrificing the technical matter. Other so-called technical courses in computer science are also being designed in this manner.

3.6 Design of everyday lecture artefacts

In a liberal arts setting, the nature of examples and exercises used in day-to-day lectures also requires careful attention. Computer science, like most other disciplines in the sciences, has suffered from indulging its students in exercises and examples from within the discipline. In our view of computer science as a liberal art, where possible, we encourage the use of examples and exercises that are taken from diverse disciplines, especially focusing on non-science areas like archaeology, linguistics, economics, environmental studies, etc. We are currently in the process of designing a formal multi-year study on the use of examples and assignments in computer science courses.

3.7 Breaking rigid boundaries

Often, a curriculum is packed tightly with a well thought out design of topics and their prerequisites. In addition, each course is packed tightly with materials related to a particular topic. We believe that it is more advantageous to cover less material in a particular topic in exchange for connecting the topic more thoroughly to other areas in, and out of, computer science. For example, consider a course on data structures. Normally, ideas such as parallelism would not (or could not) be mentioned in such a course. However, by breaking down the rigid, and often artificial, boundaries between topics, we believe that concepts learned by students will be more solid and well founded. This methodology does require faculty to communicate closely to ensure that no big ideas are missed. Faculty should also be encouraged to be extremely opportunistic about bringing ideas from their own research into the classroom.

3.8 Creating room in the curriculum

We have observed that several middle to upper-level undergraduate computer science courses at larger universities (especially those with Masters and PhD programs) are often cross-registered into their graduate program's offerings. For example, in many universities, a single offering of courses on *theory of computation*, *operating systems*, *compiler construction*, etc. exists in which both graduate as well as undergraduate students enroll at the same time. This has led us to reexamine our own set of middle/upper-level course offerings. For courses that we feel are essential to the knowledge of an undergraduate computer science major, we have included them in our curriculum. We have eliminated most courses that are required to be taken by graduate students and are not necessarily considered essential for an undergraduate degree. This has two significant effects: first,

it creates *room* in the curriculum for creating newer, more innovative courses; second, for the traditional courses that we do include, we design their curriculum tailored more towards an undergraduate level, as opposed to an offering that is required to also satisfy the knowledge requirements for a graduate-level course.

3.9 Flexibility in designing a major

What set of courses constitutes a major in computer science? The Curriculum 2001 report outlines a *core* body of knowledge that every computer science major should learn [2]. The report recommends that the core be complemented with additional coursework in other areas of computer science. In our program, consistent with the college's requirements for a total of 12 courses to be taken in a major, the following program is recommended for a computer science major:

Introductory Courses: These include our own instantiations of CS1, CS2, and Discrete Mathematics.

Core Courses: Students are required to take a course on principles of computer organization, principles of programming languages, and one course in algorithm design and analysis.

Systems Courses: Students have to take a course in either compiler design or operating systems.

Electives: Five additional courses in computer science based on the student's choosing.

Senior Thesis: All computer science majors also complete a senior project/thesis.

These requirements provide tremendous flexibility to each student since they provide the freedom to choose nearly 50% of their required courses based on their own personal interests.

3.10 Minor in computer science for all

Any student majoring in any discipline can do a minor in computer science at Bryn Mawr College. In most schools, the minor (or concentration) is typically an option available only to students majoring in mathematics and natural sciences. However, with the increasing pervasiveness of computing, we have felt it essential to open the entryways into computer science for all students. The requirements of a minor in computer science are:

Introductory Courses: These include our own instantiations of CS1, CS2, and Discrete Mathematics.

Core Courses: Any two of the core courses in computer science: Principles of Computer Organization, Principles of Programming

Languages, Algorithms: Design & Practice, Analysis of Algorithms, and Theory of Computation.

Electives: Two additional courses in computer science based on the student's choosing.

3.11 Majors in emerging computational disciplines

Our program also encourages interested students to design independent majors in emerging computational disciplines like, cognitive science, computational chemistry, computational physics, bioinformatics, geoinformatics, computational linguistics, etc. Students consult with faculty advisors in designing their programs of study. The course selections are tailored to student interests. While the computer science program becomes a default home of such majors, significant participation is necessary from faculty in other disciplines as well. Our program has been successful in influencing faculty-hiring decisions in other programs to bring in faculty with interests overlapping with computer science. Occasionally, students will also take courses offered in other area institutions. We consider ourselves lucky to be situated in a region that has more than a dozen colleges and universities in the same metropolitan area.

4. SUMMARY

In this paper, we have presented several design considerations that form the basis of our evolving computer science program. As is evident, there is much more to curriculum design than instantiating a prescription of knowledge areas into various courses. The design patterns presented above are by no means exhaustive, nor even attempting to be complete or universal. While Christopher Alexander evolved the idea of patterns in the context of architectural design, and was later applied to object-oriented design, it applies equally well in the context of curriculum design. It is also in this sense that the patterns presented in this paper are to be taken as points for further discussion, rather than as a prescription for all curricula.

REFERENCES

1. Margolis, Jane & Fisher, Allan, *Unlocking the Clubhouse: Women in Computing* . Cambridge, MA: MIT Press, 2002.
2. IEEE-CS & ACM, *Computing Curricula 2001: Computer Science Volume* , available on the worldwide web at: http://www.acm.org/sigcse/cc2001. IEEE-CS/ACM, 2001.

3. Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software* . Reading, MA: Addison-Wesley, 1994.
4. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S., *A Pattern Language* . New York: Oxford University Press, 1977.
5. Bryn Mawr College Computer Science Program worldwide web page: http://cs.brynmawr.edu.
6. *Robots gone berserk: A look at robots in film* , Bryn Mawr College course web page: http://dangermouse.brynmawr.edu/csem.
7. Gabriel, Richard P., *Patterns of Software: Tales from the software community* . Oxford: Oxford University Press, 1996.
8. Shasha, D., Lazaere, C., *Out of their minds: The lives and discoveries of 15 great computer scientists* . New York: Copernicus, 1995.
9. Brooks, Rodney A., *Flesh and machines: How robots will change us* . New York: Pantheon Books, 2002.
10. Dijkstra, Edsger W., *Selected writings on computing: A personal perspective* . New York: Springer-Verlag, 1982.