COMPUTER SCIENCE CURRICULUA FOR SMALL COLLEGES

J. S. Cameron and Z. A. Karian

Mathematical Sciences Department
Denison University
Granville, Ohio 43023

Introduction

Denison University is a small, liberal arts college located in Granville, Ohio. It has a student body of approximately 2000 undergraduates and a faculty of approximately 165. Courses in Computer Science have been offered since 1969, and in 1972 it became possible to award a Bachelor's degree in Mathematical Sciences with a Concentration (minor) in Computer Science. In 1975, a full scale Bachelor's program in Computer Science was authorized, and the first degrees were awarded in 1976. The program was designed to satisfy two objectives. First, there are a number of students who are oriented towards management and are interested in the applications of computing in industrial environments. Many of these students either pursue a double major or major in one area and develop a strong background in another field. These students frequently do not need a strong mathematical background, but do need a broad exposure to a variety of applications. The Bachelor of Arts degree program was designed for these students. Our second objective was to construct a program for those students who were interested in computer science as a profession; these students would probably go to graduate school or take technical positions in industry upon graduation. For these students we provided the Bachelor of Science degree, which is a more rigorous program.

Since the degree program was authorized in 1975, the curriculum has been significantly changed, and the resulting program seems to be one which could serve as a paradigm for those small colleges which are about to undertake a program in Computer Science.

The Original Curriculum

In order to understand how we arrived at our current program, it is necessary to see the program we originally instituted. The approved proposal to award degrees in Computer Science was strongly influenced by the ACM 68 Curriculum, and consisted of the following 12 one-semester courses:

a) Introduction to Computer Science
b) Problem Solving and Intermediate Programming
c) Data Structures
d) Discrete Structures
e) Systems Design (Computer Architecture)
f) Modeling and Computer Simulation
g) Systems Programming
h) Advanced Systems Design and Programming
i) Data Base Systems
j) Programming Languages
k) Introduction to Automata and Computability
l) Numerical Analysis

It should be noted that only the introductory course, the intermediate course (Problem Solving), and Numerical Analysis had been offered with any regularity prior to 1975, although some of the remaining courses had been offered at least once in the past. Further, although permission was granted to offer these courses, there was no guarantee that all would be offered with any degree of frequency in the future. In fact, the plan was to offer the majority of the higher level courses only once every four semesters, with no immediate plans to implement at least two of the courses (i.e. Advanced Systems Design and Automata and Computability).

Even though this type of schedule seemed reasonable, certain problems surfaced rapidly. First, each advanced course had to 'stand alone', since the instructor could not assume any knowledge on the part of students beyond the intermediate level. Thus, a course in Data Base Systems would have some students who were familiar with Data Structures, some who were familiar with Discrete Structures, and some who were familiar with both or neither. The obvious solution of adding prerequisites was not feasible since many of our students do not decide to major in Computer Science until their sophomore year, and the course schedules would preclude getting the prerequisites in time to take the desired advanced courses. This resulted in setting aside a certain amount of time in each course for a quick 'review' of relevant concepts, a review which was too fast for the students who had never seen some of the material before, and too slow for those who had already been through the same review several times.

A second, less serious, problem was that a semester was not really long enough to allow the instructor in some courses to assign enough 'interesting' problems to ensure thorough knowledge of the material (e.g., Systems Programming and Design).

Finally, in at least one case, it was discovered that a course offered in Computer Science (i.e., Discrete Structures) had a considerable overlap with a course offered in Mathematics (i.e., Modern Algebra), a situation which resulted in an unnecessary drain of departmental resources.

As these, and other problems surfaced, there was a temptation to patch the program. However, a conscious decision was made to keep the curriculum stable until we had gone through a complete cycle of offerings so that the entire program could be revised at one time. This review was made in the fall of 1977, and the revised curriculum was adopted for the fall of 1978.

Current Curriculum

The new curriculum (see Appendix A for degree requirements) is comprised of a core of six courses which are offered each year and seven courses offered on a rotating basis (see Appendix C for the schedule). The core consists of:

a) Introduction to Computer Science
b) Introduction to Statistics and Data Analysis
c) Software Structures (two semester sequence)
d) Systems Programming and Design (two semester sequence)

The introductory course in Computer Science is a standard course, as is the introductory statistics course. The inclusion of statistics in the program is a recognition of the importance that statistics plays in a wide variety of computer applications.

The Software Structures course is the largest single change in the new program, and is a combination of Intermediate Programming and Data Structures. This approach has several practical, as well as conceptual advantages. First, data structures of all types are introduced in a natural manner. The concept and implementation of arrays, stacks, queues, etc. are developed as tools in the development of algorithms or computational procedures. The same or similar problems can be solved using a variety of structures and students can gain an appreciation of the impact of data and its structure on a program, which is itself a computational structure. Second, by combining the two courses, and focusing the programming applications on the data structure aspect we are able to include more, and larger, projects than was previously the case. Finally, it is a general feeling within the department that the practice of making Data Structures a separate course only serves to make the subject obscure and mysterious. It does not reinforce the concept of data-algorithm interaction, a shortcoming we are attempting to overcome.

The final core sequence, Systems Programming and Design, was made a requirement as a result of our conviction that each student should have some basic understanding of the interaction of hardware and software, as well as an understanding of how systems work. This sequence culminates with a project such as writing a small compiler or cross assembler.

The final change in our curriculum was a merging of the Discrete Structures course with the course in Modern Algebra to form a course in Discrete Algebraic Structures. This course is the prequisite for both the Automata and Computability and the Modern Algebra courses which are offered in alternate years. The remaining courses (i.e., Modeling and Computer Simulation, Numerical Analysis, Computer Architecture, Data Base Systems, and Programming Languages) are still offered every fourth semester. This allows us to offer at least two advanced courses every semester.

Staffing Considerations

The current curriculum seems to be and in fact is, a rather ambitious undertaking for a small college. A natural conclusion would be that there was a large increase in the staff of the Department of Mathematical Sciences in order to implement it. For several reasons this was not the case. First, Denison requires all students to take one-semester laboratory science courses in at least three different departments. Introductory Computer Science is one of the courses which satisfies this requirement. The staffing requirements for this course (four sections/year in 1975 and eight sections/year in 1977) had forced the department to employ individuals who were, or were interested in becoming, qualified to teach in this area.

Further, the decision to offer the Concentration (minor) in Computer Science coincided with a natural vacancy within the departement which led to the employment of a computer scientist who could teach some of the upper division courses. Additionally as vacancies occurred within the department, a conscious decision was made to hire only individuals who had some training in Computer Science. Other advanced courses (i.e. Modeling and Computer Simulation, Numerical Analysis) were taught by mathematicians/ statisticians who had sufficient interest in the field to become well qualified for these courses. Thus, when the degree granting authority was given, there was a cadre ready who had a good idea what should, and should not, be done. In practice, the net increase in staffing for the department had been one and one half since 1973, and this is largely accounted for by the increase from four to nine sections per year of the introductory course, and some additional sections of the pre-calculus mathematics.

Although there has been some staffing increase, it should not be assumed that there is only one Computer Scientist on the staff. The replacement of a mathematics faculty member (mentioned above) by a computer scientist (PhD) plus the addition of a second computer scientist (ABD) gives us two doctoral level faculty members. Also one of the members of the mathematics faculty acquired an MS in Computer Science on his sabbatical. Thus, we have three members of the faculty with advanced formal training in the field, and one other member who has minimal formal training but has been deeply involved in the computer science program for more than ten years.

Other Considerations

The history of Computer Science at Denison is marked by the constant encouragement and interest of the University Administration. The computer's use has been increasing to the point that the average student logs over eleven hours of connect time each year, the limiting factors apparently being terminal availability and computer capacity. Over 50% of our graduates have taken at least one formal course in Computer Science, and a fairly large number take courses which use the computer as a resource (e.g., Spanish vocabulary drill). It is obvious that this interest in computing was beneficial to the development of a program. Some additional evidence of the status of computing at Denison is the fact that it was selected as one of eleven universities in the country to serve as models for academic computing. The study was performed in 1976-1977 for NSF by the Human Resources Research Organization.

Our computing facilities consist of a PDP11/45 with 116K words of main memory and 120 million bytes of on line storage. We also have three magtape drives, a card reader, a plotter, and 35 terminals. These resources serve the administrative needs, the general academic needs of diverse disciplines as well as the needs of the computer science program. Since the acquisition of the PDP 11/45 system in 1973, our computing needs have exceeded the capacity of the system and we plan to have a significantly larger system for the next academic year.

Recommendations

If any small college, comparable in size to Denison, is planning to institute a degree program in Computer Science, our advice is to plan as far ahead as possible. The Concentration (minor) in Computer Science seems to be a good approach and is an attractive method of gaining Departmental expertise in an orderly fashion. Assuming that an introductory course is currently being offered, the next step could be the introduction of sequence similar to our Software Structures. Courses in either Numerical Analysis, Discrete Structures, or Automata and Computation can be taught by members of the mathematics faculty who have some training and experience in computation. This will start the transition process in a relatively painless manner. Later, a course in Assembly language programming can be introduced which could eventually lead to a sequence in Systems Programming and Design. Other electives could be added as expertise and interests develop. Even with best intentions, it is not reasonable to expect that even a minor in Computer Science can be offered without any trained computer scientists on a staff, and the first one should be employed prior to offering the Software Structures course. Further, when natural vacancies occur within a department, an effort must be made to hire a second or third computer scientist. Under no circumstances should a degree program be undertaken with fewer than two formally trained computer scientists, since it is not reasonable to assume that any one person can handle all, or even the majority, of the advanced courses with a high level of competence.

The computing facilities of a college where a computer science program is to be initiated needs to be carefully scrutinized. With the development of a program, the facilities may not be able to cope with the additional load. Estimates of increases in connect time and CPU time should be studied so that if additional equipment is needed, it can be planned for ahead of time.

During the past several years the ACM has had committees reviewing curriculum proposals and accreditation guidelines. Intentionally or otherwise, the plight of the small (fewer than 2500 undergraduates) liberal arts colleges has been ignored. We feel our curriculum does, in fact satisfy all appropriate guidelines, but we also feel that our situation is somewhat unique. We would like either for the ACM to form a committee which can develop a realistic curriculum and accreditation guidelines for small colleges or for a working committee made up of representatives from small colleges to do the job for the ACM. It is obvious that many colleges are presently entering the Computer Science field; it would be a tragedy if there is no practical guidance available from our professional society on how to do it properly.

Appendix A

Summary of Courses Required for a Degree in Computer Science at Denison University

a. All candidates must take the following core courses:

1. Introduction to Computer Science
2. Introduction to Statistics and Data Analysis
3. Software Structures (2 semesters)
4. Systems Programming and Design (2 semesters)

b. BA candidates must take at least two of the following:

1. Computer Architecture
2. Data Base Systems
3. Modeling and Computer Simulation
4. Programming Languages

c. BS candidates must take at least four of:

1. Discrete Algebraic Structures
2. Automata and Computability (Discrete Algebraic Structures is the prerequisite)

3. Numerical Analysis (Differential Equations is a co-requisite)
4. Computer Architecture
5. Modeling and Computer Simulation
6. Programming Languages
7. Data Base Systems

d. Prerequisites (not noted earlier)

1. Discrete Algebraic Structures has a Calculus I and Linear Algebra as prerequisite
2. Differential Equations (co-requisite for Numerical Analysis) has Calculus III and Linear Algebra as prerequisites
3. All other advanced courses have Software Structures II as a prerequisite

e. General Comments

1. The Bachelor of Science degree is designed for students who wish either to pursue a career in a scientific environment or to pursue an advanced degree in Computer Science. Thus, a BS major must have either an analysis (calculus) or an algebraic (Automata and Computability) sequence.

2. The Bachelor of Arts degree is designed for those whose primary interest is industrial employment culminating a management position. Many of our BA majors take either a second major or a minor in another field, frequently Economics. Although not required, it is recommended that BA candidates take at least one semester of Calculus and one semester of Linear Algebra.

Appendix B

Course Descriptions

Most of the courses listed in Appendix A, have the usual content of courses with similar titles offered at most colleges and universities. This Appendix describes three courses which, we believe, have some unique features.

a) Software Structures

Computer programs are representations of algorithms or computational procedures, and are therefore amenable to rigorous analysis. A logically correct program may not be the most efficient one in either run time or use of resources due to poor program or data structure. This course is designed to bring together the concepts of rigorous algorithm development, program atructures, and data structures for students who have had one previous Computer nscience course.

The first semester consists of an introduction to a structured language (PASCAL), the analysis of algorithms, and a formal description of standard data types such as arrays, stacks, queues, etc.. Laboratory problems are designed to get students to implement various data structures. The second semester introduces such diverse topics as polynomial algebra, sparce matrices, generalized lists, trees, graphs, sorting and searching, hashing and memory management. The laboratory exercises in most of these areas require students to develop specific algorithms or study the relative merits of various techniques.

1. Horowiz, E. and Sahni, S., Fundamentals of Data Structures, Computer Science Press, 1976.

2. Schneider, G. and Weingart, S. and Perlman, D., Introduction to Programming and Problem Solving, With PASCAL, John Wiley and Sons, 1978.

b) Systems Programming and Design

This two semester sequence examines the design and implementation of operating systems. Considerable time is spent in developing skills in assembly language programming both as an end in itself and as a vehicle for studying the organization of computer systems. The variety of systems programming tasks studied include the design and implementation of assemblers, macro processors, loaders, and compilers. A third component of the course is the study of the structure and implementation of operating systems with an examination of possible memory, processor, and information management schemes.

Systems programming is also viewed as an important source of relevant and more complicated programming problems. Students are given experience in large scale group and individual projects which emphasize sound design, implementation, evaluation, and documentation techniques.

c) Modeling and Computer Simulation

Simulation courses in most Computer Science programs deal essentially with discrete event models and the use of special languages for the simulation of these models. The emphasis generally is on the simulation rather than model construction. We feel our course strikes a better balance model design and simulation methodology by investigating the underlying mathematical or statistical structure of the model. For example, the consideration of queuing models includes a discussion of statistical implications of Poisson processes. Simulation courses offered by Operations Research departments traditionally have emphasized these ideas. Our course may be considered a hybrid of computer science and operations research simulation courses.

The general topics include a discussion of the statistical quality of algorithms for the generation of random numbers and other random variates, queuing theory and the use of GPSS in simulating queuing models, and the use of DYNAMO for the simulation of continuous models including models defined by difference equations. We have not found any one text which fits our needs and have had to rely on notes specially prepared for the course. The following texts, however all have some of the content that we need.

1. Fishman, G. S., Concepts and Methods in Discrete Event Digital Simulation, John Wiley and Sons, 1973.
2. Gordon, G., The Application of GPSS to Discrete System Simulation, Prentice Hall Inc., 1975.
3. Yakowitz, S. J., Computational Probability and Simulation, Addison Wesley, 1977.

Appendix C

Course Schedule Over a Cycle of Offerings

The core courses are taught every year. Multiple section of Introductory Computer Science and Introductory Statistics are offered each semester. A single section of Software Structures and Systems Programming and Design courses are offered starting in the Fall Semester. The schedule for the remaining courses is:

Fall 1978
Computer Architecture
Discrete Algebraic Structures

Spring 1979
Modeling and Computer Simulation
Automata and Computability

Fall 1979
Data Base Systems
Discrete Algebraic Structures

Spring 1980
Programming Languages
Numerical Analysis

For staffing purposes it should be noted that one of the two courses each semester is mathematical in nature and is frequently taught by a mathematician with some background in computing.

References

1. "Curriculum Recommendation for the Undergraduate Program in Computer Science --A working Report of the ACM Committee on Curriculum in Computer Science," SIGCSE Bulletin 9, 2 (June 1977), 1-16.

2. Curriculum Committee on Computer Science, "Curriculum '68, Recommendation for Academic Programs in Computer Science," CACM 11, 13 (March 1968), 151-197.

3. Hunter, B., Academic Computing at Denison University - A Case Study, Human Resources Research Organization, 1978.

4. Worland, B. P., "Using the ACM Computer Science Recommendations in a Liberal Arts College," SIGCSE Buletin 10, 4(December 1978), 16-19.