The Simon Fraser One Track Universal Curriculum For Computing Science

by
T.D. Sterling, Ph.D. and J.J. Weinkam, D.Sc.
Computing Science Program
Simon Fraser University
Burnaby, B.C. V5A 1S6

Almost every academic discipline has some potential applicability to any other field of endeavor we might consider. However, there are three disciplines that share the distinction that they are widely applied in virtually every field of human endeavor. In order of seniority, there are: Mathematics, Statistics and Computing Science.

We wish to consider the problems that this situation creates with respect to how these subjects should be taught to a wide selection of students and how a curriculum can be constructed to appeal to a diversity of backgrounds and needs.

The Fundamental Problem is the Variety of the Demand

Severe obstacles are faced by any mathematics, statistics, and computing science curriculum designed to satisfy the educational aims and the academic substance of the university community. These obstacles largely are the same for the three disciplines, although we shall discuss them primarily with respect to Computing Science. They appear to be:

1. Wide ranging demands are made on the content of a Computing Science curriculum. There is the increasing computerization of biological, management, and social sciences. The humanities have been drawn into the use of computers mainly through numerical taxonomy, text processing, computer produced sound, speech, and music, and the flourishing application of graphics to a wide variety of artistic productions. Engineering and physical sciences use computers, as a matter of course, as the primary computational tool.

2. Students study Computing Science for many different purposes. There are a large number of students who expect to earn a living by working in some phase of computing (as programming technicians, analysts or software engineers). There are students who wish to follow an academic and/or professional career in Computing Science. Then there are students who just want to learn something about computing to broaden their perspective.

3. Students come to a Computing Science program with a wide variety of backgrounds. In some instances, students have had introductory courses in high school. Some even have worked as programmers before coming to the university (and some of them may have achieved considerable sophistication in programming). Other students have a fair background in ancillary topics such as numerical methods and/or statistics; others may be seriously deficient in these topics with no intention of undertaking further studies in these fields.

4. Just as all other disciplines, Computing Science must strive for an economy of intellectual content and certainly for an economy in the use of staff.

The Mathematics and Statistics Models

While it is true that each University is unique and there is no universal method of fitting a Mathematics or Statistics program into the overall curriculum, two particular models stand out as typical of a large number of cases.

On the one hand, Mathematics is usually taught in a relatively rigid sequence of courses primarily designed to satisfy majors in Mathematics. At the same time a relatively small number of service courses are offered aimed at the traditional "best" customers (Physicists and Engineers). Students in every other field either have to take courses which may or may not be relevant for them (usually they are not) or do without. Most of them do without.

On the other hand, Statistics is taught primarily to users by users. Almost every department in the university may and usually does offer some statistics courses. These courses usually introduce the student to basic concepts in a more or less intuitive and/or cookbook fashion. The student usually is not prepared to either understand or work with probabilities in any rigorous way, nor are his instructors particularly knowledgeable in Statistics - except in the more narrow uses of techniques in their own work. As a consequence, Statistics is widely misused and the student is led to a cul-de-sac from which he cannot escapt unless he starts from square one. Few of them bother.

There are numerous variations of these two models but, by and large, they are widely adopted. Computing Science is still groping and in some unconscious and unconscienable way, depending on circumstances, emulating the Mathematics model here, the Statistics model there. However, there is no good reason why Computing Science needs to copy either one of them. There may be a third method by which Computing Science can be integrated into the university curriculum and, at the same time, provide a bold and new solution to the common problems besetting the three problem disciplines.

A Single Track Universal Program

One solution, frankly based on the best features culled from the Mathematics and Statistics models, is to develop a single track program, which any student in the university can enter and leave at some desired level, depending on his background and educational aims. In order to develop such a single track curriculum, it is necessary to divide the teaching of Computing Science into three basic sectors: 1) a main stream of topics that make up the "core" of Computing Science and form the pre-requisites for other courses, 2) a "techniques" part, consisting of courses in application areas, and 3) a "foundations" part consisting of courses in theoretical and practical foundations of computation, hardware, and software. By suitable succession of core and technical topics, it may be possible to furnish the university with a program that might satisfy most all demands made on the Computing Science discipline.

The Simon Fraser Computing Science Program

The strategy outlined above has been adopted to develop the experimental Computing Science Program at Simon Fraser University. Although some additional courses are needed to broaden and enrich the program, the basic four year undergraduate program has been worked out in detail and has been approved by the Faculty. The first two years are being implemented in 1973. These are the most important part of the program since the majority of student demand will lie in these two years whereas the third and fourth year courses will be taken primarily by Computing Science majors. The complete program will be in operation by 1975.

Faculty and Staff

The faculty and staff need to operate the program consistent of the following:

a) Regular Faculty with a full time appointment in Computing Science. These faculty members will teach primarily the core courses and the Computing Science Foundations courses. In addition they may teach applications courses for which they have qualifications and interest.

b) Regular Faculty with joint appointments in Computing Science and some other area. These faculty members will teach primarily applications courses related to their discipline. In addition they may teach core and foundations courses for which they have qualifications and interest.

c) Various members of the Computing Center Staff. These individuals will take part in seminars and in courses such as Applied Programming, Hardware and Software Architecture and System Measurement and Evaluation.

d) Teaching Assistants. These are students who have already gone through part of the program. They conduct tutorials, grade papers and act as consultants in the elementary courses for which they are qualified.

e) Departmental Assistant. The departmental assistant supervises and coordinates the tutorial and consulting services, assists in the preparation of course materials, and handles numerous administrative tasks.

The Curriculum

The four year curriculum, as approved so far, is shown in Figure 1. Each box represents a course and contains course numbers and the number of credit hours. Courses indicated by an M are offered by the Mathematics department. A course box in Figure 1 indicates both the level of the course and its basic area or subject matter.

The lines connecting the boxes indicate the prerequisite requirements within the department. Some courses have additional prerequisites, outside the department, which are not shown on the Figure. For example, 291 has a Physics prerequisite.

The dashed boxes indicate areas where additional courses will be proposed. These areas include advanced applications courses in the natural sciences and advanced Computing Science courses such as Construction of Compilers and Design of Operating Systems.

Note that the second year core program consists of a single course 201. The course 200 is represented as a long thin box to convey the fact that it offers a sampling of topics normally covered in 201, 301, 302, and 400. This is discussed further below.

The number and title of each course is shown in the following table:

Computing Science Program:
Course Numbers and Titles

001 - Computers and the Activity of Man
100 - Introduction to Computing
102 - Introduction to a High Level Programming Language
118 - Computing Projects in the Arts and Sciences
200 - Introduction to Software Organization
201 - Data and Program Organization
240 - Computers in the Life Sciences
250 - Computer Uses in Environmental Studies

260 - Social Implications of a Computerized Society
280 - Computation in the Humanities I
283 - Programming Languages
290 - Introduction to Digital Systems
291 - Analogue and Digital Circuits
301 - Applied Programming I
302 - Applied Programming II
305 - Computer Simulation and Modeling
350 - Information and Public Policy
351 - Computer Graphics I: Linear Graphs
354 - Information Organization and Retrieval
360 - Computation for Statistical Data Processing
362 - Educational Uses of Computers
370 - Management and Information Systems I
371 - Management and Information Systems II
380 - Computation in the Humanities
390 - Digital Circuits and Systems
400 - Hardware-Software Architecture I
401 - Hardware-Software Architecture II
404 - Computer System Measurement and Evaluation
410 - Artificial Intelligence
451 - Computer Graphics II: Advanced Graphics
491 - Computers in Real-Time Experiments

M 104 - Elementary Computational Methods
M 306 - Introduction to Automata Theory
M 316 - Numerical Analysis I
M 401 - Switching Theory and Logical Design
M 402 - Automata and Formal Languages
M 403 - Algebraic Theory of Automata

The First Semester Core

The first semester core is divided into two parts - an introduction to computing and an introduction to a high level programming language.

The introduction to a computing course introduces concepts and methods by which problems are defined, described and implemented on computing machines. The student learns principles of algorithms and their implementation through computer compatible languages. In order to facilitate the student's rapid progress through different types of machine problems, languages, and classes of algorithms this part of the course relies heavily upon three different types of machines - each one implemented through a simulation program. The student immediately starts on a study of algorithms and their implementation suitable to a relatively simple machine (basically a Turing Machine), proceeds to a one address machine, and finally to a symbolic assembler. Other topics include number representation and conversion, non-numeric data representation, formal descriptions of algorithms, specification, languages, introduction to the principles for software construction, and a limited description of hardware.

Simultaneously the student is introduced to a high level programming language. At the present time, the language taught is PL/1 because it has a wide range of relatively easy to learn problem solutions in the humanities and social sciences as well as in the hard, natural sciences. Plans are underway to broaden this course to allow each student to select the language he will learn according to his interests and goals. The languages to be offered will include PL/1, FORTRAN, ALGOL, COBOL, APL, LISP, and SNOBOL.

After completing these courses, the student can go on to take a variety of applications and foundations courses without needing any additional core courses.

The Second Semester Core

The second semester core is designed to give the student a facility in problem solution to a more advanced level. This is represented in Figure 1 by the single course 118, although there are actually two ways in which the student may meet this requirement:

a) by taking the course 118, in which the student completes 3 to 4 projects, selected from a set of prepared projects. The student must select at least one project from each of the three categories: numerical calculation, symbol manipulation, and data processing.
b) by taking one or more unit project courses entitled "Computing in . . .", which are offered under the supervision of a faculty member in the application department. Most of the courses which show 118 as a prerequisite also allow the student to select at least two of these courses instead.

After completing this course, the student is prepared to take additional applications courses which require somewhat greater sophistication in programming and problem solving ability.

The Second Year Core

The second year core represents a slight concession to a two track system. In order to accomodate the student who wants to take an advanced course in a specialized area (e.g., graphics, artificial intelligence, or system measurement and evaluation) without making a major committment to Computing Science, the course 200 has been designed to cover a range of topics from the remaining core courses to a degree which will enable the non-major to participate in these advanced courses. In addition to covering data and program organization, normally treated at this level, the course includes material on hardware and software architecture, multiprogramming, multiprocessing, time-sharing, operating systems, and synchronization of processes.

For the student who intends to commit himself more heavily to Computing Science, the mainstream core course basically covers data and program organization. This course reviews the basic organization of programs and their data structures, control languages, and input/output requirements. Advanced methods are introduced for design and implementation of large programs including the need for and implementation of modular designed programs.

The Third Year Core

The third year core consists of a two semester sequence in applied programming. This course emphasizes business and scientific systems development, maintenance and documentation of programs. Topics include use of online systems, graphic output, user consultation, program library development, maintenance and documentation, selecting an application language, system life cycle. There are also a series of assignments, tutorials, and seminars given by the faculty and computing centre staff.

The Fourth Year Core

The fourth year core consists of a two semester course in hardware and software architecture which explores functional properties of digital computer systems. Emphasis is on the operational characteristics of concern to the systems programmer. Major topics covered are: organization of main storage; machine language; design of simulators and interpreters; CPU and I/O interaction. A large system is examined in detail. The second semester of the hardware-software architecture sequence covers the in house system and its assembly language.

Facilities and Equipment

The Simon Fraser Computer Centre has a 370/155 with 2 M bytes of main memory, 1200 M bytes of on line disk storage and three high speed printers operating under OS/MVT with HASP and WYLBUR. The WYLBUR system has just been introduced and will not be used for student jobs until the accounting system is operational and more WYLBUR terminals are available. Nevertheless, turnaround in the batch for student jobs ranges from 3 to 30 minutes.

In order to accomodate the increased enrollment in Computing Science courses anticipated over the next two years, we intend to establish a separate area for the submission of student jobs. This area will include a number of WYLBUR terminals and keypunches, and access to a card reader and a high speed printer. Most student jobs arise from the elementary courses and are processed by simulators or student oriented compilers such as PLUTO, 'WATFIV,' WATBOL, etc. These will be handled by a fast batch monitor in a dedicated partition so that fast turnaround is assured.

Experience So Far

Since the Fall 1973 semester is the first semester of actual operation there is little actual experience to report. However, the level of enrollment indicates a considerable student demand for a computing science program and the results of a course evaluation of the introductory course indicate enthusiastic reception by the students.

Conclusion

The program we have outlined here appears to offer a satisfactory solution to what has been a very difficult problem. Further implementation of that program may uncover unanticipated difficulties. But, so far, the program appears to fit within the general structure of a modern university.

Our method of teaching Computing Science prevents the university from establishing this discipline as a separate department. Rather, Computing Science is treated as a degree giving program servicing the university and students for a variety of purposes. As one consequence, no strong departmental structure develops. On the other hand, because so many departments participate in the program, the needs of Computing Science have more power of persuasion than do the needs of other, more traditional, departments.

Jurisdictional disputes are best resolved either by dual listing of courses or by giving credit for a course in Computing Science, even though the course may be offered under the aegis of another department. A good example are relevant courses which are offered under the Mathematics label. The only condition that has to be met is that the mathematician teaching these courses is qualified to do so from the point of view of Computing Science.

The content of our curriculum may not suit every taste and there is no reason why it should. But the Simon Fraser model is most flexible in this respect. More theoretical minded programs may be constructed by bolstering the core and foundations sequence or more applied minded programs formed by paying greater attention to the applications areas. It may well turn out that experience will determine the optimal balance between the parts of similar programs at any university.

Acknowledgements

Our sincere thanks to Dean Robert Brown for his unstinting support for our experimental attempts.

Much that is good and successful in our curriculum dates back to long years of collaboration with our esteemed colleague and long time collaborator, Professor Seymour Pollack, who would be with us today had he not formed a stubborn attachment to Washington University.

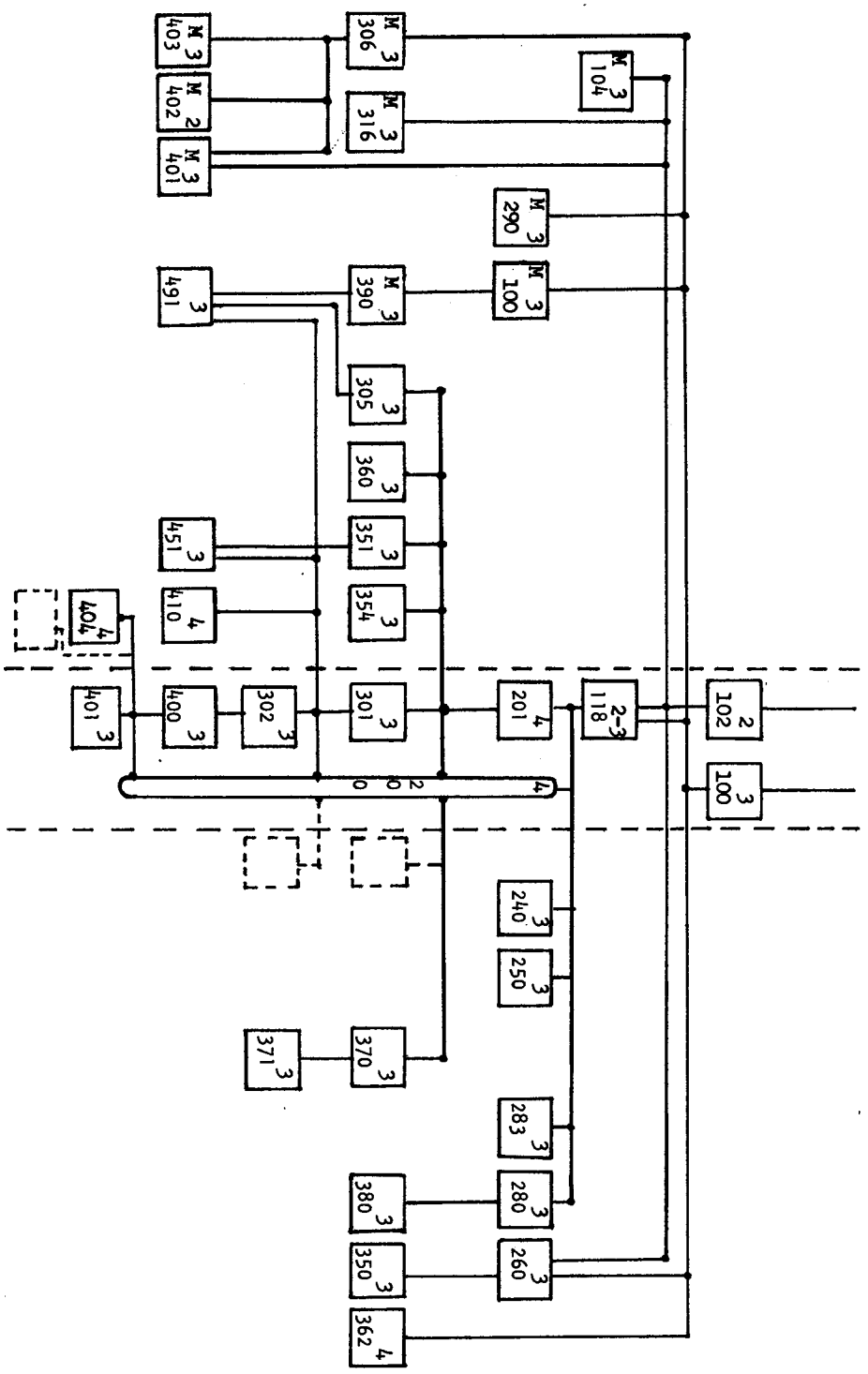

Figure 1
COMPUTING SCIENCE PROGRAM