A SET OF GOALS AND APPROACHES FOR EDUCATION IN COMPUTER SCIENCE

by: Saul Amarel

Department of Computer Science
Livingston College
Rutgers University
The State University of New Jersey
New Brunswick, New Jersey 08903

Invited paper for the 1972 Spring Joint Computer Conference.

Department of Computer Science Technical Report #10

October, 1971

Abstract

A set of goals for educational planning in computer science is presented. Among the questions discussed is the need for strong coupling between theory and practice; the importance of computer application studies, and in particular, of nonnumerical applications; the significance of establishing closer links between work in artificial intelligence and studies in the mainstream of computer design and application; guidelines for mathematical preparation in computer science; and requirements for introductory and general-education courses in the computer field. In view of the requirement for balance between educational breadth and specialization, and considering the fact that the field is changing rapidly, several approaches to long-range academic planning are presented. It is argued that a key task of computer science programs is continuous self-improvement via systematization and structuring of the knowledge in the field, and through the design of appropriate new courses.

A SET OF GOALS AND APPROACHES FOR EDUCATION IN COMPUTER SCIENCE

by Saul Amarel

1. INTRODUCTION

In order to evaluate and plan educational programs in computer science it is important to have a clear conception of the internal structure and content of the discipline, and also of its relationships with other disciplines. It is also important to have a clear set of goals, values, and priorities in the light of which given programs can be evaluated, and new approaches can be developed.

In this paper I will discuss a specific set of goals and approaches for educational planning in computer science. Much of our planning at Rutgers reflects the viewpoints that I am presenting here.

2. THE STRUCTURE OF COMPUTER SCIENCE

In [1] I have presented a framework for curriculum planning in computer science which combines a global view of the discipline and a local view. The global view is concerned with the kinds of objects, phenomena, and concepts that form the domain of discourse in computer science, and also with the pattern of relationships between this domain and other domains of study. The local view focuses on a structured description of the knowledge, problems, and activities within the discipline.

In developing the local view of the field, I find it useful to distinguish between two broad categories of activities: (1) activities concerned with problems, methods of solution and programming; and (2) activities concerned with representations, languages, schemes of processing, and system designs. They correspond roughly to computer applications and to systems. Within each of these two categories there is a broad spectrum of types of activities: development of formal systems and formulation of theories; search for fundamental principles; invention and exploration of new schemes; design and construction of new systems; and experimentation with existing systems. The range extends from purely theoretical work to practical work within the state of the art.

A. Theory and Practice

I think that it is important to maintain a view of the field which helps to minimize the distance between theoretical and practical work. Computer Science is concerned with analysis and synthesis aspects of various types of computer applications and systems. A continuous interaction between theoretical and design/experimental work is needed for a vigorous rate of progress in the field. Theoretical work in a rich applied environment is likely to receive stimulation along lines of 'relevant' models and theories. Design and experimental work in a good theoretical environment promises to lead to more efficient and powerful systems and applications.

An important reason for maintaining a strong coupling between theory and practice in academic computer science is our responsibility for guiding the growth of the discipline toward the dual goals of intellectual depth and coherence, and also of pragmatic significance for the real world of computing. Because of the phenomenal growth of the computer field, we are witnessing a rapid accumulation of vast amounts of unstructured information: specifications of many diverse hardware and software systems, records of a variety of operating experiences, discussions of different principles, techniques, and empirical advice, and many fragments of knowledge only weakly related to each other. In order to be able to build on top of what has been done in the past, and in order to understand and effectively use new developments, we must plan computer science programs in accordance with the following: (a) exposure to what is being done in the real world of computing, (b) study of ways for bringing order into what is being done, through an active search for unifying principles, general methods and theories, and (c) exploration of new computer designs and applications in the light of new concepts and theories.

Clearly any structural description of the subject matter of computer science reflects certain points of view and values about what we ought to teach and how. By not using a separate category for 'Theory' or 'Formal Systems' in a broad structuring of the field I am injecting a specific viewpoint into curriculum planning which favors 'rapprochement' between theory and practice in computer science programs.

B. Systems and Applications

Our academic planning at Rutgers is based on further substructuring of the two categories 'Systems' and 'Applications' into the four parts 'Hardware Systems', 'Software Systems', 'Numerical Applications', and 'Nonnumerical Applications' [2,3,4]. The 'Hardware Systems' part covers mainly machine organization, digital subsystems, operating systems, and design processes. The emphasis in the 'Software Systems' part is on machine level representation and programming, operating systems and language systems. The 'Numerical Applications' area is mainly concerned with problems and procedures where numerical data are dominant, such as problems in numerical analysis and in statistical processing, optimization and mathematical programming, and certain classes of modeling and simulation problems. The 'Nonnumerical Applications' area is primarily concerned with processes involving nonnumerical data. Examples of such data are: representations of problems, situations and programs; symbolic expressions; language text; abstract structures;and graphic objects. In addition, each of the application oriented areas is concerned with relevant studies in high level languages, schemes for representing problems, data bases and procedures, and related theoretical subjects.

Each of the four parts described above provides a basis for a "topic of study", which may be chosen by a student as an area of concentration. The partition into the four areas reflects an attitude to curricular design which is widely accepted at present. It has a fairly sound conceptual basis, and it leads to an implementable program which can now be supported by considerable experience and a growing literature. The conceptual basis is essentially reductionist: computer systems are decomposed in terms of different types of structural components, (structural properties determine possible modes of computation), and computer applications are differentiated in accordance with major types of data structures (properties of data structures determine types of information processes). This reductionist attitude will probably change in the next few years, as it will become increasingly desirable (and feasible) to treat hardware and software designs in a unified manner, and as the emphasis in application areas will be on broad problem types where both numerical and nonnumerical processes will play essential roles (e.g. modeling and decision making in medicine, large design and optimization problems). The changes are likely to come first

in advanced parts of graduate programs, and they will gradually move towards undergraduate programs. I expect the changes to be relatively slow, not only because of the conceptual issues involved in restructuring the field, but also because of the natural inertia of educational systems where required courses, supporting texts, and other relevant resources, cannot change overnight.

A structured description of the field is an important input for planning an educational program, and it already reflects a specific set of viewpoints and decisions on priorities. We need, in addition, further clarification of priorities in order to determine (a) the relative emphasis that the program should place on different parts of the field, and (b) the way in which individual students should be guided through the program.

Our emphasis at Rutgers is on the relationship and mutual impact of application areas and computer systems. This implies a fairly balanced set of activities in applications and systems, with major effort in problem solving methods, computer procedures, languages, systems software,and associated theories. This approach is consistent with our view that studies of computation mechanisms and system designs should grow in an environment which is rich with significant and challenging computer applications.

3. COMPUTER APPLICATIONS

I believe that serious work on new types of computer applications is essential for the healthy development of computer science. The challenges presented by new applications and the constructive attempts to meet them will continue to be key factors in the evolution of computer systems and in the conceptual maturation of the field. Work on new applications will lead to increasing collaborative ties with people in other disciplines; this will increase the 'surface of contact'between computer science and other areas, to the benefit of all involved. In this context, computer science could provide an academic home for certain important activities associated with applied mathematics: the formulation and study of models, the development of methodologies for bridging the gap between a real life problem and its representation within a system wherein the problem can be solved, and the study of broad approaches to problem solving.

In our programs at Rutgers, computer applications are studied at various levels. A freshman-level introduction to computing is oriented to hands-on experience with computer problem solving, using BASIC on a time sharing mode. Sophomore-level courses in computer problem solving focus on a variety of problem types (numerical and nonnumerical) and on the use of several high-level languages — FORTRAN, PL/I, SNOBOL. Thus computer languages and their features are introduced in the context of their use for formulating computer procedures of different types in a variety of problem areas. The study of computer organization, machine level programming, and systems software follows the courses where problem solving is the dominant theme. Thus a student is first exposed to what kinds of things computers do and how can they be used, and then he is led to consider how computer systems operate, how they are structured and how they can be designed.

The study of computer applications continues in our upper level undergraduate classes with courses in numerical methods, data processing methods and nonnumerical algorithms. At the graduate level, two introductory (and required) courses on numerical analysis and nonnumerical algorithms are followed by a collection of courses and seminars which explore in different degrees of depth a variety of application areas, procedures, and approaches to problem solving.

Recently we established an (NIH supported) research resource on computers in biomedicine. This provides a focus for collaborative activities with people in medicine, bioengineering, psychology and ecology on a broad range of problems that involve the use of computers in biomedical inquiry. This type of research activity is an extremely important environment for advanced graduate work and for faculty research in various aspects of advanced computer applications.

A. Nonnumerical Applications

The study of nonnumerical applications is of central importance in a computer science program. It provides insight into the broad notion of computers as symbol manipulators, and it permits a clear view of many key information processing concepts that become obscured when presented only in the context of mathematical algorithms for numerical problems. Also, the domain of nonnumerical applications is extremely large, and it contains a great variety of significant problems in the field. In addition to the broad spectrum of problems from elementary data processing to complex decision and problem solving in artificial intelligence, the entire area of systems software (languages and operating systems) is mainly founded on nonnumerical processes and techniques.

B. Artificial Intelligence

Work in artificial intelligence problems provides an excellent opportunity for the study of nonnumerical processes and for the clarification of certain basic issues in software design. It is important to work toward the development of closer links between artificial intelligence studies and other studies in the mainstream of computer applications and of systems design. I believe that this will be of benefit to computer science education, and in general, it will stimulate new, significant developments in the computer field.

It is becoming increasingly clear that there is no sharp dividing line between the problem solving procedures of artificial intelligence and the procedures of today's 'conventional' software (see [5]). It is useful to think that computer procedures lie in some sort of continuum, and each procedure in the continuum is characterized by the relative amount of systematic versus heuristic knowledge available to it, as well as the degree to which this knowledge can be efficiently exploited by the procedure for the solution of specific problems in its domain. At the one end of the continuum, where systematic knowledge is dominant and its mode of utilization is highly efficient, we have the customized procedures and the formally validated algorithms of the well developed computer application areas. At the other end, where amount of systematic knowledge is low, we have the general, flexible, goal directed procedures of artificial intelligence, where a set of relatively weak problem-specific principles are combined with heuristic methods for the control of processes that search for solutions. In between, we have the bulk of procedures that run on computers at present — including procedures that manipulate computer languages, and those that control computer operations.

The notion of a procedure continuum, where the emphasis is on types of information processes and on types of knowledge associated with them, promises to provide a fruitful conceptual framework for education in computer science. It would be interesting to explore it further in the context of future curriculum designs.

C. Numerical Applications

Numerical analysis and several other subjects in the general area of numerical applications are among the best developed in the field in terms of systematic knowledge available and theoretical treatment of algorithms. I think that these subjects have an important place in a computer science curriculum, not only because of their great significance for many computer applications, but also because they provide models of 'high quality' procedures and of successful theoretical approaches to their study. Such models are needed for progress in other, less developed areas of the field.

4. INTERACTION WITH MATHEMATICS

The study of numerical applications is closely related to mathematical studies. A strong mathematical background (especially in calculus and linear algebra) is needed for work in most numerical applications. Conversely, numerical applications are in themselves significant topics of study in mathematics.

The question of mathematical preparation for work in computer science must receive careful consideration in the development of computer science programs. Theoretical work in computer science relies on several branches of mathematics and logic. In addition to the formal background needed for numerical applications, various topics in abstract algebras, in logic and foundations, in combinatorial analysis, in graph theory and in probability theory are needed for work in well developed areas of nonnumerical applications and of computer system design.

A problem arises when we consider the training of people who are oriented to the bulk of design and application activities in the field. For the most, these activities are in areas where relevant theoretical knowledge is poor. There is considerable concensus among computer science educators that people working in these areas must have at least a certain level of 'mathematical maturity'. This does not imply necessarily the knowledge of specific mathematical facts and methods. What is meant usually is training in disciplined modes of thought, in the manipulation of abstractions, and in the development and use of methods that can apply in a great variety of situations. To obtain this type of training, we have several approaches: (a) provide sufficient exposure to offerings in regular mathematics curriculi, (b) restructure some mathematical offerings to attain within a limited time, both the 'maturity' objective and the objective of education in mathematical topics of special relevance to work in the computer field, and (c) develop computer-based courses that address themselves directly to the underlying cognitive skills implied by 'mathematical maturity', without using conventional mathematical material. These approaches are not mutually exclusive. I believe that it is an important challenge for computer science education to develop the approach (c). In parallel, I would place considerable stress on the approach (b). I am basing this on the conviction that even those who graduate at present from a computer science program and expect to take a programming job which does not require an immediate specific knowledge of mathematics, must be equipped against rapid obsolescence in a field which is exgremely dynamic and which requires continuous learning and adaptation to new

concepts and techniques. As theoretical knowledge grows in their area of activity, they must be prepared to understand it and to use it. Formal approaches will continue to develop in computer application areas; they are likely to penetrate more and more in systems design areas as concern with quality and efficiency of designs will continue to grow, and as methods of system analysis and evaluation will become increasingly available.

Wallace Feurzeig and his colleagues at Bolt Beranek and Newman have started to design courses where computer programming is being used as a key to the development of certain basic mathematical ideas and of skills for abstract reasoning [6]. This work has been oriented so far to pre-college instruction. It would be extremely interesting to develop courses of this type for introductory college instruction. In addition to their effectiveness for the development of 'mathematical maturity', such courses would be exceptionally well suited for the introduction of computers to a large number of college students who, because of their 'fear of math', hesitate to take introductory computing courses in their present form (which is commonly biased toward mathematical calculations).

5. INTRODUCTION TO COMPUTING AND TO GENERAL EDUCATION ABOUT COMPUTERS

Computer science departments have an important responsibility to provide broad introductory computing courses to the entire College population. In order to reach a very wide group of students, with a great possible variety of backgrounds and orientations, it is important to develop courses with no special mathematics requirements and with the main emphasis on the essentials of information processing and on hands-on problem solving experience with computers. The design of such courses is of considerable value for computer science as a new discipline, since it forces attention on fundamental concepts in the field and on imaginative ways of communicating them.

One of the most promising areas for the use of CAI in a college curriculum is in introductory courses in computer science and in mathematics. It would be highly desirable to stimulate the development of such computer-based courses in computer science departments; this would be a valuable activity for both faculty and students.

In our program at Rutgers we have had some experience with an introductory course where the emphasis was on uses of computers and on their impact on society. We found that students at the freshman level were much more interested in what

computers are and how they can be used, rather than in general questions about the impact of computers on society and the broad significance of a growing computer culture on people. I think that it is profitable and important to discuss these issues in a senior seminar; at that level, students have more knowledge (of computers and of other things), and more maturity and motivation to study relationships between computers, man, and society.

6. SUGGESTED APPROACHES TO LONG RANGE EDUCATIONAL PLANNING IN COMPUTER SCIENCE

The question of educational breadth versus specialization is central to long range academic planning in computer science. As the computer field grows, there is an increasing demand for in-depth training of a variety of specialists. If the rate of generation of general principles and methods in the field will continue to lag behind the rate of production of specific computer systems and specific application packages, then there will be a tendency for computer science curriculi to be overwhelmed by a variety of fragments of detailed, specialized material — and for students of computer science to have negligible opportunity of being exposed to other disciplines, and to areas of intellectual activity where computers do not necessarily play major roles. On the other hand, it is essential for computer science not to grow in isolation, but to actively seek and strengthen intellectual bonds and working contacts with other disciplines. Accordingly, future contributors in the field must develop the ability to commnicate and collaborate with people in other areas, and to be sensitive to ideas and concerns outside their discipline. This implies the need for balance between activities within the discipline and outside it. Also, the nature of studies in computer science should reflect the dynamic character of the field, and the need to develop in students a capacity for independent learning and growth.

The following set of approaches are suggested in response to the requirements for educational balance and for adaptability to fast changes in the field:

(a) The core of computer science courses that are needed for an undergraduate major should be limited to between a third and a half of the total courses needed for an undergraduate degree. These ratios will be much higher at the graduate level. In any event, students at all levels should be encouraged to take courses in other areas.

(b) An important part of academic planning is to develop (in some detail) options for combined studies involving computer science and other disciplines. Undergraduate areas of specialization should be created by combining the computer science core courses with courses in other appropriate disciplines (in particular, with Mathematics, Electrical Engineering, and Management Sciences/ Business Administration). This implies a limited core curriculum offered by the computer science unit, and close curriculum coordination with the other academic units. Similar curriculum coordination should develop in graduate programs. One of the most effective ways of stimulating contact with other disciplines at the graduate level is to facilitate/induce projects and research which are directly related to advanced computer applications in these disciplines.

(c) An appropriate balance (and coordination) should be established between the lecture courses, the seminars, and the project-oriented courses that are components of a program. Lecture courses should concentrate on fundamentals, on structured presentations of parts of the field, and on general guidance to independent study of the growing body of 'raw data' concerning specific computer systems, languages, and applications. Seminars should focus on specific new developments in the field (exploratory current research, attempts to develop principles and theories) and on discussions of broad issues (e.g. computers and society). Project-oriented courses should provide opportunities for synthesis of information obtained from various sources, for collaboration (with people in the field, as well as with others), and for extensive practical work within the state of the art. An important project course would be an undergraduate, upper-level "design studio" where students would work on substantial software designs of operating systems, language processors, interfaces, etc.

(d) The interface between undergraduate and graduate programs in computer science should be flexible — with students moving relatively freely both ways, and courses moving steadily from the graduate to the undergraduate program. This movement of courses does not mean an increase in the number of undergraduate courses, but a restructuring, updating and general strengthening of the undergraduate program — whose overall size should remain relatively stable. One of the reasons for a flexible undergraduate — graduate interface is the fact that most students entering graduate computer science programs still come from different academic disciplines, and their knowledge of computer science varies widely. As undergraduate computer science becomes more widespread, the nature

of Masters-level programs will change, and prerequisites for graduate admissions are likely to become more demanding in areas of computer systems and applications. In the meantime, criteria for graduate admissions should remain more flexible on questions of specific subject matter; the emphasis should continue to be on overall undergraduate academic performance (especially in mathematical sciences), personal characteristics such as initiative and inventiveness, and motivation to work in computer science. Because of the dynamic character of the computer field, I expect that we will have for some time a process of curricular changes where the Masters program of today will be essentially included in the Bachelors program of tomorrow. In general, a terminal Masters program in computer science will continue to provide the training needed for advanced professionals in the computer field. Increasingly, Bachelor programs in computer science will also train many of the professionals in the field.

(e) A major part of advanced graduate work in computer science should be devoted to exploratory projects and to research. A good research environment is essential. This means an appropriate intellectual climate, good communications, and easy access to computer resources. An important goal, both for faculty and for graduate students, would be to introduce order and cohesiveness in the field, to search for basic principles and general methods, and to design better academic programs in computer science. As part of these activities, new courses should be developed, with the emphasis on fundamentals and also on ways of using computers as integral parts of the preparation and administration of the courses. It would be desirable to have doctoral research develop in the context of such educational projects.

In general, we should recognize that systematization of fundamentals in the computer field, and design of appropriate academic programs is a key responsibility of computer science educators. These activites constitute a process of continuous self-improvement of computer science programs. Such a process is an essential part of any academic program; it is of special significance in computer science. Efforts toward academic planning and program improvement should receive support and recognition — the same as direct teaching and other types of research and development. Contributions in these areas will have important implications both for the effectivenss of computer science education, and also for the overall quality and efficiency of work in the computer field.

REFERENCES

[1] S. Amarel, 'Computer Science : A Conceptual Framework for Curriculum Planning', Communications of the ACM, Vol. 14, No. 6, June, 1971.

[2] 'Undergraduate Program in Computer Science'; Department of Computer Science, Livingston College, Rutgers University, New Brunswick, N.J. 08903; April 1971. Available from the Department.

[3] 'Graduate Program in Computer Science'; Department of Computer Science, Rutgers University, New Brunswick, N.J. 08903; July, 1971. Available from the Department.

[4] 'Descriptions of Graduate Courses in Computer Science'; Department of Computer Science, Rutgers University, New Brunswick, N.J. 08903; July, 1971. Available from the Department.

[5] S. Amarel, 'Problem Solving and Decisions Making by Computer : An Overview', in 'Cognition:A Multiple View', P. L. Garvin (ed.), Spartan Books, N.Y., 1970.

[6] W. Feurzeig et al., 'Programming Languages as a Conceptual Framework for Teaching Mathematics', Report No. 2165, June 30,1971, Bolt, Beranek and Newman, 50 Moulton St., Cambridge, Mass. 02138.