CE EDUCATIONAL COMPUTING - BEYOND AN INTRODUCTION

by

William J. Rasdorf, P.E.*

INTRODUCTION

The rapid advances occurring in interactive microcomputing and computer science have provided the engineer with powerful means for processing, storing, retrieving, and displaying data and have thus made computer science an essential part of every engineering discipline. Applications of existing computer science technologies are spreading and are giving engineers a sophisticated means of rapid access to a wide variety of information, solutions to complex problems, and ways to model complicated engineering systems. Applications of new computer science technologies in such areas as databases and artificial intelligence are also being achieved [Fenves82, Fenves84, Sriram83]. Using computer science effectively in engineering applications is recognized by many as the key to increased individual, company, and national productivity.

In the future, an integrated combination of computer-aided analysis and design techniques will need to be developed for all types of civil engineering design problems. This will require applying computer science principles and practices to analyze a variety of civil engineering systems in order to determine their response to external influences. The implications of this requirement for the academic community are clear: we must prepare our students to use computer methods and applications as a part of their fundamental education.

The low cost and ready availability of microcomputers have removed the financial barrier that limited the use of computers by many small and medium sized design and construction firms. In addition, the microcomputer software industry has made available a growing supply of useful engineering and administrative programs. Satisfying the special needs of civil engineering organizations, however, often requires in-house software development. As a result, civil engineering firms are seeking graduates whose engineering training has included a study of the fundamentals of computer science. Unfortunately, the number of graduates who have received this education has not kept pace with the demand, and has therefore limited the number of qualified applicants to civil engineering firms.

* Assistant Professor of Civil Engineering and Computer Science, North Carolina State University, P. O. Box 7908, Raleigh, NC 27695.

The potential of interactive computing, computer-aided design, and computer graphics in civil engineering has been demonstrated. It is now the responsibility of colleges and universities to prepare their students to make use of this potential. By providing instruction in and use of contemporary computing fundamentals in their academic curriculum, colleges and universities can improve the professional qualifications of their civil engineering graduates. Although the academic curriculum must be upgraded to meet this challenge, many engineering schools are currently making the changes necessary to enhance their students' knowledge of the fundamental concepts of computer science. One method of doing so is by adding a new course beyond the introductory language course.

Purpose

Emphasizing computer science as an integral part of an engineering student's education depends on many factors - educational practices, student needs, school and university computer acquisition plans, faculty support, research needs, available funds, and so on. This paper addresses two of these issues: educational practices and student needs. It then describes the general structure and content of a new engineering computer course and discusses those aspects of computer science that would enable civil engineering students to perform effectively in their subsequent classwork, in graduate school, and in their careers.

The ideas presented here may guide educators who wish to expand their curricula while maintaining the emphasis of the course content within departmental boundaries. The paper will also be of interest to engineering firms that are evaluating the educational profile of new graduates to locate those who possess, in addition to their engineering skills, both the ability to evaluate and use production software and the ability to organize and supervise the development of in-house software.

PRESENT EDUCATIONAL PRACTICES

The traditional, undergraduate civil engineering curriculum has generally contained an introductory course that teaches the use of a programming language. More often than not that language has been FORTRAN, although Basic, Watfiv, and PL/1 have also been commonly used. At North Carolina State University, the Civil Engineering department offers the introductory course at the sophomore level. Fundamental objectives of the course are to teach students to learn to use a programming language (FORTRAN 77), an operating system, and a text editor. Additional goals include teaching students to represent the logic of the problem-solution process by using flowcharts or algorithms, to write programs that efficiently implement the logical representation, to compute resulting values with varying input, and to communicate their solution plan to other engineers who may perform the necessary calculations.

Unfortunately, the results of this educational experience are often limited: students learn to draw flowcharts and they learn the mechanics of a programming language. The novelty of the concepts and the unforgiving rules of language syntax, combined with unfamiliar editors and long debugging sessions, leave the student with little time to achieve all of the goals of the course. As a result, many fundamental topics remain only partially discovered or completely unaddressed. Fortunately the faculty can, by establishing an appropriate choice of objectives, effectively incorporate these topics into a follow-up course.

OBJECTIVES

Some of the essential objectives of a new course emphasizing computer science principles and practices for civil engineering students should include enabling them to:

- independently develop computer programs for solving engineering and scientific problems;
- use existing application software and library subroutines;
- understand the use of computer-aided design, interactive analysis and design, and optimization techniques;
- be familiar with the use of state-of-the-art computer hardware including micro, mini, and mainframe computers, terminals, printers, plotters, and data acquisition units; and,
- be familiar with the use of state-of-the-art software including operating systems, text editors, text processors, and electronic mail and other communications systems.

Of particular importance are the needs of students with respect to developing new and using existing computer programs. Regarding new program development, students should have a thorough working knowledge of at least one scientific programming language, be able to use the computer to solve any problem requiring computation in any course in which they are subsequently enrolled, and be able to verify that their program is working properly and providing correct answers. When using existing programs and subroutines that apply to the problems and topics covered in civil engineering courses, students should be able to:

- select the correct program for the problem at hand,
- understand the associated user manual,
- verify the accuracy of the program by using test-case problems that have known solutions, and
- be able to interpret the results of the program.

Accomplishing all of these objectives may at first seem insurmountable. However, a careful analysis of departmental objectives and resources may illuminate many steps that can be taken to achieve them.

A Note of Caution

Two guiding principles should be considered when establishing a program to satisfy the objectives mentioned above. First, the civil engineering student is not majoring in computer science. Although computer science is emphasized, it must be remembered that the student's primary goal is to learn the principles and practices of civil engineering. Other aspects of his education should support that goal. In that regard, the emphasis on computer science should assume a decided, but subordinate, role in the educational program.

The second principle is that a computer should be used only when it is advantageous to do so. Iterative problems, data management problems, solutions to sets of simultaneous equations, and graphic design problems are examples of applications that can be efficiently solved using a computer [Holtz79]. However, arriving at solutions to some problems will remain a manual process. Civil engineers should readily recognize when manual calculations are more appropriate than using the computer.

REINFORCING AND ENHANCING COMPUTER USE

Two approaches can enhance the civil engineering student's knowledge of computer science. The first approach is to enable the student to continually use, in his civil engineering courses, the knowledge obtained from his introductory computer science course. Doing so requires that the faculty encourage regular and frequent use of the computer. The engineering curriculum should be structured so that it continually builds on the introductory computing course by integrating computer-aided design and analysis techniques into upper-level courses.

While learning the fundamental principles of engineering, students will have frequent opportunity to apply computational skills in completing homework, laboratory, and special project assignments. Programming solutions to problems in all of these different settings should be encouraged. To develop the student's computational dexterity, faculty should develop assignments that require the student to develop original programs in addition to using existing programs.

The second approach to increasing the civil engineering student's knowledge of computer science is to introduce a second course that stresses the advanced computing methods and applications specific to civil engineering. Such a course should emphasize advanced programming language topics, data storage and use, and program organization and control.

Related Approaches

Writing programs is only one of many aspects of a civil engineer's computer education. Two other noteworthy areas of educational emphasis have been suggested. In [Appleton83] Appleton suggests a course that emphasizes civil engineering applications rather than programming techniques. Sommer suggests an even broader computer-aided engineering education composed of three distinct systems: the instructional system; the problem-solving system; and the experimental system [Sommer82]. Used as an instructional system, the computer provides information about a particular engineering subject - including text, figures, and tables - and prompts the student with questions and exercises. As a problem solving system, the computer offers the student an engineering problem, sets of possible steps to its solution, and justification for each step in the solution process. The student obtains the solution by learning the correct sequence of steps. In an experimental educational mode, the computer can be used to monitor experiments and acquire and analyze data, or it can be used to simulate experiments without the use of laboratory facilities [Sommer82].

Each of the above aspects of computer use lies well within the realm of civil engineering educational objectives; each merits attention. However, the focus maintained in this paper is on advanced programming techniques and their relationships to engineering problem solving techniques.

COURSE CONTENT

Proper computer program development requires that the student have a sound knowledge of software engineering - the application of language-independent engineering principles and practices to the development of large computer programs. Topics of primary importance are data structures, program control, and program organization.

Data Structures

The study of data structures is too often incomplete because instruction in using higher level data structures, including tables, lists, trees, graphs, networks, etc., is usually neglected. As a result, engineers are often unaware of the conceptual basis for these data structures, of the availability of algorithms for processing them, and of the means whereby they can be implemented in FORTRAN. This instructional neglect is of particular concern because of the many engineering applications that can be represented by higher level data structures. Networks and network processing techniques, for example, can be used in finite elements, framed structures, hydraulic systems, transportation systems, construction scheduling, and surveying. Regardless of the application a thorough civil engineering education requires that the student be able to recognize the type of data structure that most precisely represents the problem he is modeling and that he be aware and capable of using the facilities available for processing the data.

The underlying motivation for emphasizing data structures is the important role that data plays in the design process. The design process is highly data generative. It begins by the designer determining the values for basic data items that reflect early design decisions. These include describing the structure's configuration (topology and geometry [Rasdorf83]) and the applied loads. From this basic data additional data is derived by the designer and by application programs, and is stored in a database. An analysis program, for example, generates forces, moments, and displacements using geometric and loading data.

When all of the data needed to represent the structure has been derived, the design is complete. Its final, external representation consists of a combination of project specifications, design calculations, and drawings, each of which can be obtained in an automated manner with a computer-aided design system - a system designed to aid in developing a computer representation of the complete specification of a structure or facility. The role that data plays in the design process is extensive, and its proper representation and use is an important ingredient in the success of computer-aided design systems. It is therefore imperative that the civil engineering student understand this role and the relationship between program data structures and the design process data items they represent.

Program Control

Structured programming is the process of developing and organizing computer programs by using well-structured control constructs. Control in programs is the means of directing the flow of operations and is achieved by sequential execution, iteration, recursion, conditional execution, and selection. All computer languages provide a built-in set of constructs to directly implement some, all, or combinations of these forms of control [Hughes78, Fenves79]. The engineering student should fully understand each form and be able to build equivalent structures from what is available in his language if the appropriate forms do not already exist. The proper use of programming control constructs is necessary for developing efficient and well-structured programs.

Program Organization

Program organization stems from structured programming. Control constructs play a role in program organization, but subprograms are the major organizational component. The proper combination of subprograms, as well as prudent selection and use of local and global variables, must be carefully considered by program developers. Each program should be divided into well-defined modules, each with a clear purpose and specific interfaces to the remainder of the program. Such practices can readily be achieved if the engineering graduate recognizes them to be a direct outgrowth of an organized approach to the more global problem solution process. Just as data structures reflect the composition and development of the physical structure, programs must accurately reflect its evolution through the engineering design process.

Other Concerns

A variety of other topics should be included in the proposed course. Among them are problem-oriented languages, user-program interfaces, software engineering principles, basic machine operation, defensive programming, debugging techniques, and documentation.

Students should be made aware, for example, that good documentation means much more than inserting internal program comments after a program has been written. Comprehensive documentation includes developing user manuals and technical documentation manuals, as well as inserting internal program comments during program development. The similarity of these practices to documenting design calculations and developing engineering reports should be pointed out.

Another example, defensive programming, is a means of dealing with unexpected or unanticipated program operation. The unexpected invariably leads to errors, and debugging is required to eliminate them. Practice with and familiarity of the programming language, combined with using a good compiler, will expedite the removal of syntax errors. On the other hand, removing semantic errors which deal with the meaning and logic of the program is more difficult. These are most efficiently removed by using of special debugging programs provided by the operating system. To become accomplished at using these debugging programs, students can be required to debug with them as an integral part of their programming assignments in the second course.

Programming Assignments

Student homework assignments that involve writing computer programs should relate directly to a civil engineering problem. Ideally the assignments should demonstrate an analytical solution process, incorporate programming control and organization concepts, and integrate data structuring concepts.

Each programming assignment should be designed to include a reasonable number of new computing concepts and practices. It should provide the student the opportunity to practice the new programming concepts presented in the lectures, handouts, and text by implementing them in programs. For example, a student should learn to understand the usefulness and scope of good documentation by writing the documentation himself. However, despite the fact that comprehensive documentation is mandatory in all civil engineering programs, it is not necessary to require its incorporation into every programming assignment. Such repetition reduces the time available for the student to spend on new assignments and may negatively influence his opinion of the repeated material.

Judging Student Performance

Programming assignments alone are not a sufficient measure of student performance. Without additional incentive, students tend to concentrate on coding and debugging, ignoring lecture material and

helpful programming techniques that require additional thought and practice. To ensure that the concepts presented in the lecture are well understood, non-programming assignments based on the lecture notes, handout material, and text are imperative. Periodic tests covering all of the material presented in the course will reinforce the material.

CONCLUSION

Despite the fact that computer science is not the business of civil engineers, a great deal of knowledge in computer science can benefit our profession. The ASCE Computer Practices Committee of the Technical Council on computer Practices summed up the engineer's responsibilities related to computing in [Beck79] by saying:

> "The engineer and his firm accept the professional responsibility, liability, and risks associated with computer-aided design when they affix their seal to drawings and specifications. Therefore, to ensure reliability, the engineer must develop competence in the use of judging computer programs through proper education and training."

I have suggested that one way of conveying the essentials of computer science to future engineers is through introducing an additional computing course into the civil engineering curriculum. Indeed this is a first step, and such a course should be a part of a broad effort aimed at enhancing the civil engineering computing education. Developing a program to achieve this objective and incorporating it into the Civil Engineering curriculum is a topic worthy of further discussion.

ACKNOWLEDGEMENT

Portions of this paper were based on the report "Computer Utilization by Undergraduates in Civil Engineering" by the Committee of Undergraduate Programs of the Department of Civil Engineering at North Carolina State University [NCSU82a]. This report was released in February of 1982. The author wishes to acknowledge the helpful efforts of the committee members.

The computer methods and applications course described in this paper evolved from a similar course taught by Professor Steven Fenves of Carnegie-Mellon University, an innovator of computer use and applications in civil engineering. His guidance and support are appreciated.

REFERENCES

[Appleton83] Appleton, J. H. and Miller, E. T., "Teaching C. E. Students to Use Computers," Proceedings of the Eighth Conference on Electronic Computation, American Society of Civil Engineers, Pages 172-181, February, 1983.

[Beck79] Beck, C. F., Committee Chairman, "Ethical Considerations In Computer Use," Journal of the Technical Councils, American Society of Civil Engineers, Volume 105, Number TC2, Pages 415-427, December, 1979.

[Fenves79] Fenves, S. J. and Schiffman, R. L., "Quality Assurance of Engineering Software," Journal of the Technical Councils of ASCE, American Society of Civil Engineers, Volume 105, Number TC1, Pages 57-74, April 1979.

[Fenves82] Fenves, S. J. and Rasdorf, W. J., "The Role of Database Management Systems in Structural Design," Proceedings of the IABSE Colloquim on Informatics in Structural Engineering, International Association of Bridge and Structural Engineers, October, 1982.

[Fenves84] Fenves, S. J., Maher, M. L., Sriram, D., "Knowledge Based Expert Systems in Civil Engineering," Proceedings of the American Society of Civil Engineers, April, 1984.

[Holtz79] Holtz, N. M., "The Need for Research and Education in Computing in Civil Engineering Design," Unpublished, 1979.

[Hughes78] Hughes, C. E., Pfleeger, C. P., Rose, L. L., Advanced Programming Techniques. A Second Course in Programming Using FORTRAN, John Wiley and Sons, New York, NY, 1978.

[NCSU82] "Computer Needs for the School of Engineering," North Carolina State University Engineering School Computer Committee Report, 1982.

[NCSU82a] "Computer Utilization by Undergraduates in Civil Engineering," North Carolina State University Department of Civil Engineering, Committee on Undergraduate Programs Report, February, 1982.

[Rasdorf83] Rasdorf, W. J., "Relational Database Modeling of Building Design Data," Proceedings of the Third Conference on Computing in Civil Engineering, American Society of Civil Engineers, April, 1984.

[Sommer82] Sommer, H. T., Werner, B. D., Howie, G. R., "Computer-Aided Engineering Education," Mechanical Engineering, Pages 38-40, December, 1982.

[Sriram83] Sriram, D., Maher, M. L., Fenves, S. J., "Applications of Expert Systems in Structural Engineering," Proceedings of the Conference on Artificial Intelligence, April, 1983.

CE EDUCATIONAL COMPUTING - BEYOND AN INTRODUCTION