AC 2007-1478: INTRODUCING CIVIL ENGINEERING ANALYSIS THROUGH PROGRAMMING

George List, North Carolina State University

George List is Head of the Civil, Construction, and Environmental Engineering Department at NC State University

Introducing Civil Engineering Analysis through Programming

Abstract

This paper describes a course in computer programming that is being offered to freshmen and sophomores in civil engineering at NC State. Visual Basic (VBA in Excel) and MATLAB are being used as the programming languages. Much of the learning occurs through reverse engineering and imitation. Typical civil engineering problems are used to present the programming concepts. Especially in the instance of VBA, students learn how to combine the use of spreadsheet functions with VBA code. The paper includes an overview of the course and examples of the materials covered and the teaching techniques employed. General thoughts are also presented about the directions in which programming education may be headed in the future.

1.0 Introduction

Courses about computer programming have been part of undergraduate curricula for more than half a century. For example, the electrical engineering department at CMU was teaching computer programming in FORTRAN in the late 1960's as a way to introduce logical thinking (e.g., flow charts) and programming skills. Other disciplines adopted such courses more slowly. Consequently, the topic of this paper is not new.

As Rasdorf [1] indicates, in the late 1970's, civil engineering programs began to embrace the idea of including computer programming classes in their undergraduate curricula. The argument was, in part, that "students must be prepared to use computer methods and applications as a part of their fundamental education. It is the responsibility of colleges and universities to incorporate contemporary computing fundamentals into their academic curriculum to improve the professional qualifications of their engineering graduates. These graduates will in turn be able to provide their increasingly important expertise to both the engineering profession and the academic community."

Today, while ABET (see the 2007-2008 criteria, for example) does not explicitly require a course in computer programming [2], it is clear that ABET expects students to learn computer programming skills. Criterion 3, focused on program outcomes and assessment, stipulates that engineering programs must demonstrate that their students attain eleven outcomes including "an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice." Moreover, Criterion 8, which deals with program criteria, indicates that: "Programs must provide opportunities for students to learn the use of modern engineering tools. Computing and information infrastructures must be in place to support the scholarly activities of the students and faculty and the educational objectives of the program and institution."

Even though more than half a century has passed since computer programming was first taught to engineering students, the academic community is still engaged in debate, innovation, experimentation with ways to accomplish that objective. Courses like calculus, physics and chemistry have not seen nearly the same degree of transformation. Computer programming classes have been taught by computer science departments, engineering departments, and special teaching teams. The languages used have ranged from PAL, FORTRAN, WATFIV, PL1 and

ALGOL to Pascal, Basic, Visual Basic, C++, JAVA, Mathcad, MATLAB, and, undoubtedly, many others.

Moreover, it seems that the perceived need for such classes has risen, fallen, and risen again. Not to make a major issue of this point, at first, it was necessary to write a computer program to do any kind of analysis. General application programs like Excel did not exist. Then software products advanced, and products like LOTUS-123, Quatro, and Excel emerged, and the perceived value of learning how to write programs declined. Instead, students were taught how to use specific packages, like Excel and AutoCAD, but not how to create programs from scratch. There was a sense that such programs would be sufficient for solving engineering problems. The pendulum shifted back when it became apparent that off-the-shelf software was not going to be able to address every problem solving need.

However, it soon became apparent that of-the-shelf software was not going to address every problem that might arise. Moreover, with the advent of software packages like Excel with VBA and MATLAB, it became apparent that these products could be used to teach logical, algorithmic thinking. So the excitement about teaching computer programming returned; and innovation continues. New ideas continue to be developed. This paper presents a recent realization of a computer programming class that has been developed to meet the undergraduate needs at NC State in civil engineering.

2.0 The Evolution of Computer Programming Classes

From an historical perspective, computer science departments began offering classes in computer programming in the late 1960's. Brady [3] provides a very early description of such courses, ones developed at three universities between 1964 and 1970. As an aside, in 1968, the author took a class in computer programming taught by the electrical engineering department at Carnegie-Mellon University. Gruener and Graziano [4] review introductory courses in computer programming that were created before 1978 while Austing, Barnes, and Engel [5] provide a more comprehensive survey of papers focused on computer science education published during the 1960's and 1970's.

In civil engineering, Rasdorf [1] identifies papers as early as 1982 that focused on teaching students how to use computers. He also identifies an unpublished document by Holtz from 1979 that espouses the need for research and education in computing in civil engineering design. It could be that civil engineering did not begin to require courses in computing until about this time.

In the years since 1980 considerable attention has been devoted to computer programming classes for civil engineering undergraduates. This paper is hardly the first. For example, Hart and Groccia [6] describe a 1994 freshman course that focuses on "fundamentals in civil engineering and computers." The course "incorporates computer application skills, the development of oral and professional presentation skills, team teaching, small group cooperative learning. Christensen and Rundus [7] describe a 1998 core engineering class that "teaches the use of computers for solving engineering problems. The course is innovative in its teaching of not only a high-level programming language, but also a mathematics package, spreadsheet, and formal design methods." Al-Ansari and Senouci [8] describe a 1999 course that uses Mathcad as a

teaching and learning tool, in the context of designing concrete footings. They assert that "Mathcad, which possesses efficient computation and presentation capabilities, holds strong potential as a teaching tool and learning aid for education and training."

The impact of flexible, graphically adept analysis packages is clear. Schumacher, Welch, and Raymond [9] in 2001 focus more specifically on command and control in describing an introductory course in programming in which "The LEGO Mindstorm robots are used … to teach fundamental computer programming concepts and introduce the concepts of autonomous vehicles, embedded computer systems and computer simulation." Palazo and Phillips-Lambert [10] describe a 2003 similar effort at the University of Memphis that adds instruction in Visual Basic so the students see a second programming environment. Jewell [11] describes a 2001 class in which equation solvers are used to teach concepts of hydraulic design for water distribution systems and open-channel flow. "Students then develop the nonlinear mathematical model for a simple example, solve the model using an equation solver, and check the correctness of the solution. Students are able to investigate the dynamic response and the sensitivity of the model by varying the equation solver input variable values. Next they apply the theory and solution methods to a practical applications exercise. The final step is to complete a comprehensive, realistic design problem." Westphal, Harris and Fadali [12] focus on a more general level of thinking related to translating natural language problem descriptions into computer code. Their experience is that "using LabVIEW and Alice as graphical foundations, with several carefully designed examples, may help students more quickly learn the process involved in computer-based problem solving than they would with traditional techniques." Bowen [13] describes an introductory class in computing that is focused on MATLAB as a replacement for FORTRAN. As Bowen observes, "Inclusion of computer programming early in the curricula has been seen by the Civil Engineering faculty as a way of improving the students' skills in logical reasoning, application of technical knowledge, and quantitative problem solving." The students "write MATLAB programs as an integral part of a structural design project where groups of students compete against one another to produce a truss-style balsa wood bridge having the highest profit. Throughout the semester a series of homework assignments require students to write MATLAB programs that calculate separate bridge characteristics that determine the cost and benefit of their design, such as amount of wood used, number of bridge nodes, bridge mass, and estimated strength." Barry and Webb [14] describe a similar course, focused on teaching numerical methods to engineers, which uses trusses, electrical circuits, explosions, tsunamis, and other phenomena to teach ideas like interpolation, integration, regression, and solutions to non-linear differential equations.

Other papers, that were part of the ASEE Conference in 2001, present information about recent trends in the structure and content of computer programming classes. Clough, Chapra and Huvard [15] describe classes at three very different institutions that have similar characteristics: emphasis is placed on "engineering problem solving, elementary numerical methods, and algorithmic programming. Software vehicles include Mathcad, MATLAB, and, in particular, Excel and its VBA programming language. Use of a traditional, stand-alone programming language, such as Fortran or C/C++, is postponed beyond these introductory courses." Hertiner and Scott [16] describe a program in which MATLAB was adopted as the basis for teaching programming skills, especially for electrical engineers. Litkouhi and Pritchard [17] discuss a similar situation in which a course in "Computer Programming for Engineers" was based on

Excel, Visual Basic, and Mathcad. The course is described in more detail by Naraghi and Litkouhi [18]. In a similar sense, Haering [19] describes the use of Excel in sophomore-level engineering mechanics courses. Head *et al*. [20] present a different perspective in which their university elected to employ C++, in a cleverly controlled environment, as the basis for an introductory course in computer programming.

It is clear that this paper is not the first that has been written on the subject. It is also not the first that has focused on using VBA and MATLAB as programming languages. The new ideas in this paper relate more to the way in which VBA and MATLAB are being used, the way the course is structured, and the manner in which the material is being taught.

Of course, these innovations, those being presented, and those that have been discussed, have all occurred in the context of more general curricular reform, something that is very important to keep in mind. Porter *et al*. [21] describe the innovations in undergraduate education that occurred at NC State in response to the need to "evolutionize" the freshman and sophomore courses for a cohort of 1100 students per year. Sack *et al*. [22] portray the critical need to transform civil engineering undergraduate programs and the challenges that that presents. More recently, Grigg *et al*. [23] discuss the merits of integrating IT (information technology) into civil engineering curricula and the issues that ensue. The transformation process is far from over. This author thinks there will be far more change in the next fifty years than there has been in the last fifty. And the evolution of computer programming classes is only a small part of that change.

3.0 The New Class

Presently, at NC State, civil engineering students are expected to take CSC116, an introductory class in computer programming taught by the computer science (CS) department. It uses Java as the programming language and emphasizes topics important to computer science majors: control structures, classes, methods, data types, object-oriented programming and design, graphical user interface design, etc. It is not that this class is "bad" in any way, but it simply does not meet the needs of the civil engineering undergraduates. It teaches "irrelevant" material at the "wrong" level. These drawbacks are compounded by the fact that the classes are a combination of CS and non-CS students, which means the non-CS students frequently struggle to achieve good grades. The pace is too quick and the material too complex and "irrelevant" from their perspective.

A couple of years ago, the department experimented with teaching its own computing class, an experiment that went well, but heavily taxed scarce teaching resources and detracted from the department's ability to offer graduate courses. So the experiment ended.

When the author became a member of the department in 2005, one idea brought to his attention was the need for a better programming course. Through a series of discussions it became apparent that this might provide a way for the department head to interact with some of the undergraduate students, so a decision was made to experiment again.

Interestingly, it was not just the civil engineering department that was thinking about teaching its own computer programming class. Mechanical engineering and industrial engineering also thought they might do the same thing. Moreover, they had similar ideas about how the class

should be structured: use Visual Basic (VBA), inside of Excel, as the primary programming language, supplement it with MATLAB, and place a heavy emphasis on example problems, drawn out of the disciplinary environment, so the students gain a sense of how learning to use these tools might have a long-term value for their careers. The choice of VBA was driven by a sense that it might be used in practice, since Excel is so common, it is very approachable and transparent, as in the debugging features it affords; and MATLAB was deemed valuable for research. In civil engineering, a strong sentiment existed that if such programming skills were common among the undergraduates, there would be opportunities to do new and creative things in upper division classes insofar as lab assignments, homework exercises, and undergraduate research experiences were concerned.

The particulars of the current course are as follows. It is a full semester in duration, with about fourteen lectures in programming concepts and fourteen labs. Homework is given weekly. VBA, in Excel, and MATLAB are used as the programming environments. The problems are drawn out of civil engineering as examples of applications that might arise in practice or research. The students are held responsible for reading assignments, lab reports, homework, and tests.

During the initial weeks of the semester, the students study VBA programs that have been prepared by the instructor or the TA's. These examples illustrate important programming concepts and show how code can be written efficiently and effectively. None of the programs can be regarded as "toys". They are often simple, but they address real-world problems in a real-world manner. The degree of complexity varies widely, from simple, straight-forward, sequential computations to iterative procedures, recursive procedures, event-based simulation, etc. The longest program, focusing on simulation, comprises 18 pages of code. The students study these programs to learn by example, or "reverse engineering" the details of writing code. They "take the programs apart" and figure out how they work. They are asked to answer questions like "what happens in lines 10-15", "why is the code the way it is in the For..Next block", "What is the purpose of Subroutine X or Function Y?"

A transition to student-written programs occurs as the semester progresses. It starts with the students developing programs during the lab sessions. This is followed by homework exercises in which the students develop their own code. Careful monitoring of student progress and confidence dictates how fast the transition occurs.

The lectures cover basic principles and illustrate programming concepts. The labs cover details, walk students through programs, and provide a forum for answering questions. The instructor is present for both the lectures and the labs. At least two additional TA's are present in each of the labs. The students work in pairs, to help them learn better, and quicker, and to facilitate the creation of study groups.

Programming concepts that are covered include variable types, naming "conventions", subroutines and functions, built-in functions, the passing of parameters, For..Next loops, Do..Loops, If..Then blocks, Select..Case blocks, and recursive procedures. The intent is to teach students programming skills that they are very likely to use if and when they do write code.

The initial labs focus on structural analysis, like computing forces and moments in trusses, to reinforce concepts learned in statics, a class that many of the students have taken or are taking at the same time. Later labs focus on geotechnical engineering, transportation engineering, water resources, environmental engineering, project scheduling, and engineering economics.

In one of the labs, students are asked to create VBA code that duplicates calculations done by formulas in the Excel spreadsheet. The exercise helps them see the correspondence between the way they would enter formulas in Excel and the way they have to write VBA code to do the same thing. This also helps them verify that their code computes the intended results.
The initial programs are ad-hoc in structure, like early FORTRAN programs used to be. That is, they simply start, without much overhead, and progress through a series of processing steps, creating variables as needed, to generate a set of answers or results. Later programs are more formal and resemble production-quality code. However, formal programming practice ideas like revision documentation, data base structures, error trapping, and input-output controls are not stressed. Those that become interested more seriously in programming are encouraged to take formal programming classes offered elsewhere on campus.

The labs that focus on project scheduling are particularly popular because the students can see the purpose of the code. They can see what it is doing. They can also see the relevance of the code to construction and find excitement in studying critical paths, resource constraints, etc. Both deterministic and stochastic project scheduling problems are examined. The combination allows the students to see, from a project scheduling standpoint, why some tasks might be on the critical path only some of the time. It also makes it possible to talk about uncertainty, stochasticity, and random variables.

The problems in engineering economics are presented toward the end of the semester. One of the problems is coded in both VBA and MATLAB so the students can see the similarities and differences in how the two programming environments address the same computational tasks. The manner in which MATLAB handles vector-matrix algebra is also compared with the way in which such calculations are accomplished in VBA.

A detail, which is probably not insignificant, is that the class is well supported by staff resources. Besides the instructor, there is a graduate teaching assistant (TA) and two to three undergraduate teaching assistants. At least three people are present during each of the labs and help the students complete the assignments. For 40 students in 20 pairs, this means one person for every 6-7 “lab stations”.

In summary, the class has a heavy emphasis on teaching programming skills that the students are likely to use. It does not bog them down with formal training in programming. It also tries to teach them by reinforcing what they have learned, and adding to it, so they never get lost.

The 2006 fall semester class had 60 students. Only a couple of them had taken CSC116. Some had taken a one-credit introduction to programming class. There was one lab section with 40 students and another with 20. Most of the students were sophomores. A few were juniors or seniors. Formal reviews were conducted but have not yet been processed and their feedback has not been received. The anecdotal feedback from students and advisors that have talked to

students who took the class is very positive. The spring semester class has 80 students in two labs of 40 students each. Only one student has taken CSC116. More students wanted to take the class, but enrollment was limited. Enthusiasm seems high and the first labs have gone very well. One challenge that remains is to determine how the class can be expanded from 80 students to 170, the number of undergraduates per year that would need to take the course to graduate if CSC116 was not an option.

4.0 Things to Improve

Even though the initial feedback is encouraging, many things could be improved about the class. One would be to expand the portfolio of lab problems. A greater collection of problems, drawn from all areas of civil engineering, would help stimulate greater interest and excitement, and ensure that the interests of all students are captured. Additional homework problems would also provide more material to reinforce the concepts being taught. More MATLAB problems would also be useful, including problems that duplicate the VBA solutions. This would let the students see how the code is different depending on the programming environment employed. Programs that have built-in mistakes (bugs) would also be helpful so that students learn how to find and fix bugs.

5.0 What Lies Ahead?

In closing, what lies ahead, for the new NC State class and for computer programming courses in general? The answer, of course, is not clear or certain, but generalizations are possible.

First, such classes are likely to be in civil engineering programs for some time. This is not only because of their value in teaching students how to create computer programs but also because they teach logical thought, analytical procedures, and algorithmic thinking.

Moreover, there will “always” be problems that professionals need to solve that are not handled by off-the-shelf software, or situations where off-the-shelf software is too expensive or takes too long to learn, to justify the investment, compared with writing some code. It is also likely that programming courses for engineers will “always” be informal, as opposed to formal, in the way they teach programming, because they need to focus on “need to know” topics as well as process/ processing logic and algorithmic thinking. Even if the “code writing” expertise is not explicitly used, the “logical thought” skills will be used in finding creative ways to solve problems.

It also seems likely that reverse engineering, or learning by example, will increase in popularity, because it helps to ensure that the students do not get lost, that they do not learn bad habits, and that they do not get frustrated because they are wasting time trying to do things they are not equipped to do.

In closing, the future of computer programming classes seems bright. The course at NC State is meeting a critical need; and it seems well matched to the needs of the students. It is likely to continue to evolve and mature in the future. More generally, civil engineering departments will continue to face similar situations, where they need to devise or revise programming courses so

they meet student needs. So the trend towards similar courses seems assured. As the references indicate, many different realizations of the "same" idea have emerged in the past. It seems likely that they will continue to emerge, with each realization being slightly different, tailored to the needs of a particular department, set of students, etc. Integration of these ideas by academics and practitioners will make all civil engineering programs better, and help the profession do a better job of meeting the needs of its customers, in practice and research.

REFERENCES

[1] Rasdorf, W. J., "Computer Programming in the Civil Engineering Curriculum," *Journal of Professional Issues in Engineering*, 111:4, pp. 141-148, October 1985.

[2] Accreditation Board for Engineering and Technology (ABET), *Accreditation Criteria for Engineering Programs, 2007-2008*, Baltimore, MD, 2007 (http://www.abet.org/forms.shtml).

[3] Brady, A. H., "The Introductory and Service Courses in Computing: Some Experiences and a Critical Assessment," *ACM SIGCE Bulletin*, 2:2, pp. 31-36, June-July 1970.

[4] Gruener, W. B., and S.M. Graziano, "A Study of the First Course in Computers," *ACM SIGCSE Bulletin*, 10:3, pp. 100-107, August 1978.

[5] Austing, R.H., B.H. Barnes, and G.L. Engel, "A Survey of the Literature in Computer Science Education Since Curriculum '68," *Communications of the ACM*, 20:1, pp. 13-21, January 1977.

[6] Hart, F.L., and Groccia, J.E., "Fundamentals in civil engineering and computers-a freshman course," *in* Lawrence P Grayson (ed), *Frontiers in Education Conference Proceedings: Twenty-Fourth Annual Conference*, San Jose, California, November 2-6, 1994.

[7] Christensen, K., and D. Rundus, "A First Course in Computing for Engineers," *Proceedings of the 1998 ASEE Southeastern Section Meeting*, pp. 247-255, April 1998.

[8] Al-Ansari, M. S., and A. B. Senouci, "Use of Mathcad as a Teaching and Learning Tool for Reinforced Concrete Design of Footings," *Computational Applied Engineering Education*, 7, pp 146-154, 1999.

[9] Schumacher, J., D. Welch, and D. Raymond, "Teaching Introductory Programming, Problem Solving and Information Technology with Robots at West Point," *Proceedings of the Frontiers in Education Conference*, Volume 2, pp. F1B – 2 to 7, 2001.

[10] Palazolo, P., and A. Phillips-Lambert, "Divide and Conquer: Teaching Programming from a Visual Perspective, *ASEE Southeast Section Conference Proceedings*, 2003.

[11] Jewell, T.K., "Teaching Hydraulic Design Using Equation Solvers," *Journal of Hydraulic Engineering*, 127:12, pp. 1013-1021, December 2001.

[12] Westphal, B.T., F.C. Harris Jr., and M.S. Fadali, "Graphical Programming: A Vehicle for Teaching Computer Problem Solving," *Frontiers in Education Proceedings*, 2003.

[13] Bowen, J. D, "Motivating Civil Engineering Students to Learn Computer Programming With a Structural Design Project, *ASEE Annual Conference & Exposition: Engineering Education Reaches New Heights*, Salt Lake City, UT, June 20-23, 2004.

[14] Barry, S.I., and T. Webb, "Multi-disciplinary Approach to Teaching Numerical Methods to Engineers Using Matlab," *ANZIAM Journal*, 47, pp. C216-C230, 2006.

[15] Clough, D.E., S. C. Chapra, and G. S. Huvard, "A Change in Approach to Engineering Computing for Freshmen – Similar Directions at Three Dissimilar Institutions," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[16] Herniter, M. E., D. R. Scott, and R. Pangasa, "Teaching Programming Skills with MATLAB," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[17] Litkouhi, B., and P. J. Pritchard, "Freshman Engineering Courses at Manhattan College - Lessons Learned," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[18] Naraghi, M. H.N., and B. Litkouhi, "An Effective Approach for Teaching Computer Programming to Freshman Engineering Students," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[19] Hearing, W., "Integrating Computer Tools into Sophomore-Level Engineering Mechanics Courses," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[20] Head, L., J. Kay, J. Schmalzel, G. Arr, C. Foster, S. McDermott, M. Sterner, K. Whelan, and J. Wollenberg, "Building Confidence and Skills: A Prep Course for Computer Programming," *Proceedings of the2001Annual ASEE Conference*, Albuquerque, NM, June 24-27, 2001.

[21] Porter, R. L. L. J. Bottomley, M. C. Robbins, W. V. Yarbrough, S. A. Rajala, and H. Fuller, "Introduction to Engineering Problem Solving - A New Course for 1100 First Year Engineering Students," *Proceedings of the 1999Annual ASEE Conference*, Charlotte, NC, June 20-23, 1999.

[22] Sack, R., R.L. Bras, D.E. Daniel, C. Hendrickson, K.A. Smith, and H. Levitan, "Reinventing Civil Engineering Education," *Frontiers in Education Conference Proceedings*, 1999.

[23] Grigg, N.S., M. E. Criswell, D. G. Fontane, and T. J. Siller, "Information Technology in Civil Engineering Curriculum," *Journal of Professional Issues in Engineering Education & Practice*, 131:1, pp 26-31, January 2005.

Biographical Information

GEORGE F. LIST: Dr. List is presently the Head of the Department of Civil, Construction, and Environmental Engineering at North Carolina State University. He is a graduate of Carnegie Mellon University (BSEE, 1971), the University of Delaware (MEE, 1976), and the University of Pennsylvania (Ph.D., CE, 1984). He is a past Vice-Chair and Secretary of the ASCE Department Heads Council Executive Committee.