The Freshman Engineering Problems and Programming Course: Integrating New and Old Tools

Lawrence Genalo
Iowa State University, genalo@iastate.edu

Brian D. Williams
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/mse_conf

Part of the Engineering Education Commons, Other Materials Science and Engineering Commons, and the Science and Mathematics Education Commons

Recommended Citation
http://lib.dr.iastate.edu/mse_conf/35

This Conference Proceeding is brought to you for free and open access by the Materials Science and Engineering at Iowa State University Digital Repository. It has been accepted for inclusion in Materials Science and Engineering Conference Papers, Posters and Presentations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Abstract
At Iowa State University in the early 1980's the first engineering course, Engineering Problem Solving, began incorporating FORTRAN programming in an integrated, three semester credit course. At first this course was taught in "dry lab" style with the students assigned to find keypunch terminals outside of class in order to create and edit their programs. Soon after that the course switched to interactive VAX terminals and the first classroom/laboratory was born. This room allowed for a terminal for each pair of students and in-class programming assignments. The engineering problems content was also "dry-labbed" at first. For example, in order to learn beginning statistics students were presented with a table of data from an "experiment." These "experiments" were most often hypothetical and found at the back of a textbook chapter.

Disciplines
Engineering Education | Other Materials Science and Engineering | Science and Mathematics Education

Comments
This is a conference proceeding from Proceedings of the ASEE Annual Conference 1 (1995): 1532. Posted with permission.
**Session 2353**

**The Freshman Engineering Problems and Programming Course: Integrating New and Old Tools**

Lawrence J. Genalo, Brian D. Williams  
Iowa State University

**Introduction**

At Iowa State University in the early 1980's the first engineering course, Engineering Problem Solving, began incorporating FORTRAN programming in an integrated, three semester credit course. At first this course was taught in "dry lab" style with the students assigned to find keypunch terminals outside of class in order to create and edit their programs. Soon after that the course switched to interactive VAX terminals and the first classroom/laboratory was born. This room allowed for a terminal for each pair of students and in-class programming assignments. The engineering problems content was also "dry labbed" at first. For example, in order to learn beginning statistics students were presented with a table of data from an "experiment." These "experiments" were most often hypothetical and found at the back of a textbook chapter.

After switching the programming environment to in-classroom terminals, the method of teaching all topics, programming and problem solving, changed to incorporate the computer as a learning tool. Using early graphics editors, lessons for the course topics were created on-line and used as an electronic blackboard during lectures. This had the added advantage of creating a complete set of course notes that was accessible to all students on the network. The problem solving content of the course was still a "dry lab" with the only change being the data was now listed on-line rather than in the text. There was clearly a need for more hands-on engineering work to enliven the theory of problem solving topics and to create more meaningful, real-world programming scenarios.

By the late 1980's, the department switched to a networked microcomputer environment and discarded the old VAX environment. All new lessons were needed to maintain the on-line courseware aspect. Microcomputer software tools, such as spreadsheets and word processors, needed to be introduced. For the first time students could have the possibility of taking home not only the programming tasks, but also the lessons. In surveys of incoming students, a growing percentage of students had microcomputers available in their residences and had some experience using them. The Engineering College at this time joined the Synthesis Coalition, one of the first funded by the National Science Foundation, and attempted to attain goals for undergraduate engineering education which included increased teamwork and industrial practice, more hands-on experiences for students, synthesizing interdisciplinary content, improved communication and social context skills, and employing technology to assist the learning process.

This course, as the gateway to the engineering curriculum, was facing mounting pressures to do more, include more, and produce more, with no change in the number of credits devoted to it. Achieving a greater level of integration - integration of engineering problems with programming, integration of computational tools for problem solving, and integration of new technology with traditional engineering practice - has helped to alleviate these pressures while creating a motivational engineering experience at the doorway to the curriculum. The following section will describe an extensive courseware development effort that has resulted in the ability to divert some "lecture" time to more active and collaborative learning time. The integration of traditional hands-on engineering practice and new technology will be described in the last section of this paper.

**Courseware Development**

As the computer revolution progressed at Iowa State University, the need arose for more ways to integrate the computer into the classroom. The idea of using a computer as a lecture aid came to mind. From this great idea came the need to develop software containing the lecture notes in an interactive form for usage in the classroom. Macromedia's Authorware Professional software package was chosen to complete the task. Under the guidance of faculty members, the lecture material was developed. The development is a fairly long process.

Once the material to be programmed in has been
obtained, the programming begins. Developing an Authorware program is a lot like creating a flowchart. Different icons are available on a menu bar. Each icon serves its own purpose. There are eight basic icons. A display icon is used to present text and graphics, such as a picture of a relevant newspaper article that has been scanned into the computer. An animation icon can perform small basic animations on anything on the screen. For example, an animation of how the forces in a statics problem are applied is very helpful to a student.

Erase icons are used for erasing things that have been displayed using a display icon. Wait icons are used to specify a certain amount of time or a specific activity that must happen before continuing through the program. Wait icons are useful to allow the user to control the flow of the program. Of course, decisions must be made in any flowchart and Authorware programming is no different. Decision icons are used to branch to different parts of the program depending on a certain condition. Interaction icons, along with calculation icons are the true heart of an Authorware program. By combining these two icons and a little creativity, almost any program can be made interactive. Map icons are for layering the program. In other words, the map icon is a group of other icons. Besides these basic icons, there are two flags and three special icons.

Flags are used to start and stop the program. Therefore flags are usually only of use to the developer when testing an application. The other three icons consist of a movie icon, a sound icon, and a video icon. These icons play movies, sounds, and display frames from a videodisc player. Using these icons, the student developers work with the faculty members to achieve the best possible final product. There are many questions to ask when programming in Authorware. Will the program hold the student's attention? Will the student be able to use it on his or her own? Is the appearance of the software appealing (or at least not boring)? The smallest details can make the difference between a package that will work and a package that will only hinder further integration in the classroom. For example, the color schemes used in courseware are important. They must be exciting enough to be interesting and yet visible in a darkened room using an LCD projection system. Black and white just won't do the job.

Once the software has been developed, it is then tested by the faculty. If everything checks out okay, it is then loaded onto the network. When it has been loaded, the program is available to anyone with access to the computer labs. Currently the courseware available on the network consists of lessons, tutorials, and tests.
The tutorials are interactive practice questions to be used by the student outside of class time. For every lesson, there is a tutorial to go with it. The tests are exams which have been given in class in previous years. The tests are also very interactive. This allows the student to get extra practice working problems before an exam. The lessons, tutorials and exams are updated on a yearly basis. Students can also buy a copy of the lessons that have been printed in a spiral bound book. One major advantage of having a printed copy is this allows the professor to cover the material faster which leaves more time for hands-on experiments. The professor can cover the material faster because the students can look over the material before class and they don't have to take as many notes in class.

From talking to students about the class, it is easy to see they like having the option to do extra problems on specific lessons or before a test for review. When a student uses a tutorial or test on the network, the intent is to have the student learn while having fun. The interaction is more than showing the student how to do the problem. The student selects a topic and the computer will ask questions about that topic. When the student gets done working the problem and types in the answer, the program will give feedback. If the student has the answer correct, the program announces that. If not, the program will let the student try again or after a few tries, the computer will give the answer to the student. The answer is not just handed to the student. The computer shows how to work the problems and points out where the student may have made their mistakes. This is invaluable to the student and the professor. The student gets an explanation that is very close to how a professor would explain it without actually having to track him down. The professor can use the problems in class as examples or even assign them as in class exercises.

Obviously these programs are valuable to more than just students at Iowa State University. The NEEDS database allows the programs to reside on the Internet so anyone with access to the Internet can get the lessons, tutorials, or exams. The process of getting these programs on the NEEDS database is not difficult. A few simple forms are filled out (one of these forms is filled out on Mosaic on the Internet and mailed) and your program is uploaded to a NEEDS database site. The program will then be put into the correct directory of the NEEDS server after it is cataloged. This is all it takes to get the program on a NEEDS server, so everyone and anyone who wants to use it can get to it. The NEEDS servers can make class time even more efficient. If a student asks a question and the professor doesn't know the answer, the professor might be able to find a program dealing with the topic, download it, and use it as a teaching aid. The NEEDS servers are also important research sites for students when doing hands-on experiments, which will be discussed in the next section.

Integration of Hands-on Experiments

At the same time that the courseware development program has been ongoing, the introduction of hands-on experiments to illustrate the engineering problems topics proceeded. The first such experiment involved teams of students experimenting with a manual wheelchair. One student team member was weighed and placed in the chair with the brakes locked. Other student team members then attached a cable connected through a pulley system to a weight platform. Student team members than placed weights on the platform until the chair "just" began sliding across the floor. By using various student teams performing the same experiment, a table of chair/occupant weights versus platform loads was developed. The students were encouraged to question the data, the assumptions, and the experimental methods employed. On several occasions, at the insistence of the students, data points and experimental conditions were rechecked.

This data was then used for many applications problems throughout the semester. When graphing was studied, the students plotted their own data and noted data points that seem to be inconsistent with the pattern developing. When statistics were studied, the students calculated the "best" model for their data from linear, power, and exponential choices. When spreadsheets were introduced, the students again employed statistical and graphical methods to analyze their data. When statics was introduced, a free body diagram for the wheelchair experiment was drawn. In all cases the students demonstrated their willingness to take ownership of the data and question its engineering sensibility. The Authorware lessons, delivered in a high-tech learning classroom provided the theory and first examples of how to apply that theory, but the hands-on experiment and its analysis brought that theory to life. The students, anxious to apply analysis tools to their own data, need little instruction in spreadsheet use to get started. Without the motivation of the hands-on experiment, the spreadsheet introduction becomes just another academic exercise (read drudgery) to be performed.

Another hands-on experiment that has been introduced deals with resistor values. A sample of resistors with some nominal resistance is selected for analysis. Each student measures the actual resistance with an ohmmeter and compares it with the color-coded nominal resistance.
Again, a table of values is created. This time students write programs to calculate the basic statistics (mean, median, standard deviation) associated with the sample. Once again, a greater degree of student ownership was seen. Measurement questions were raised about some of the data points. The percentage of resistance values falling outside the nominal range was discussed. It was enlightening to see some students who thought it would be "impossible" for a commercially purchased resistor to lie outside the acceptable range come to grips with the experimental evidence.

The integration of computer tools (both student programs and canned packages), engineering problems topics, hands-on experiments, and computer-aided-instructional courseware has proved fruitful in this course. Time has been taken from previous lectures and put to use in more active and collaborative learning experiences. The use of computer tools is motivated by the need to solve engineering problems just as it would be in practice.

Acknowledgments

The authors wish to thank the National Science Foundation, the Synthesis Coalition, Iowa State's College of Engineering, and the Division of Engineering Fundamentals and Multidisciplinary Design for providing support for this work.
LAWRENCE J. GENALO

Lawrence J. Genalo obtained the B.A. degree from Hofstra University in 1971, and the M.S. and Ph.D. degrees from Iowa State University in 1974 and 1977, respectively. His Ph.D. is in Applied Mathematics with an emphasis in Systems Engineering. He has been at Iowa State University since 1971, and has been an Associate Professor since 1981. He has served ASEE as Program and Division Chair for Freshman Programs and DELOS. His current research interest is in bringing high-technology classroom delivery systems into greater use in engineering education through his work with the NSF-funded Synthesis Coalition.

BRIAN D. WILLIAMS

Brian D. Williams is an undergraduate student at Iowa State University pursuing a B.S. degree in Computer Engineering. Mr. Williams is involved in design, development, and testing of course software that is used in engineering courses at ISU. He has been involved with this project and the Engineering Fundamentals and Multidisciplinary Design department since early 1993. Currently Brian is on a co-op working as a Software Test Engineer for Microsoft Corporation. Mr. Williams will return to ISU in the fall to continue his education.