DOCUMENT RESUME

ED 410 921 IR 018 483

AUTHOR Elliot, Noreen
TITLE Are Computer Science Students Ready for the Real World.
PUB DATE 1997-00-00
NOTE 6p.; In: Association of Small Computer Users in Education (ASCUE) Summer Conference Proceedings (30th, North Myrtle Beach, SC, June 7-12, 1997); see IR 018 473.
PUB TYPE Reports - Evaluative (142) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *College Students; Communication Skills; *Computer Science; Employment Opportunities; Higher Education; *Job Skills; Problem Solving; Professional Education; Professional Occupations; Stress Management; Teamwork; Technology; *Thinking Skills; Time Management; Undergraduate Study; *Workplace Literacy

ABSTRACT

The typical undergraduate program in computer science includes an introduction to hardware and operating systems, file processing and database organization, data communication and networking, and programming. However, many graduates may lack the ability to integrate the concepts "learned" into a skill set and pattern of approaching problems that would enable then to fit easily into a business organization and to succeed in such an environment. This paper describes the nontechnical and non-theoretical skills necessary to survive and advance in the workplace, and suggests activities to develop those skills. General skills required in the workplace include the ability to: manage time and set priorities; work effectively as part of a team; generalize learning to various situations; manage stress; communicate effectively; and support ideas and problem solutions. "Computer" skills required in the workplace include: logical thinking; ability to eliminate program errors; reuse software; adjust to constantly changing project specifications; and develop good design skills. To succeed in the field, students must learn how to effectively apply knowledge they learn to the real world. They must also be able to communicate ideas effectively, work with a variety of people, and expand their knowledge and skills as the field changes. (SWC)

Are Computer Science Students Ready for the Real World

Noreen Elliot
Senior Training Analyst
Equifax Marketing Services
P.O. Box 740006
Atlanta, GA 30374-0006
(770) 740-4708

Abstract:

The typical undergraduate program in computer science includes an introduction to hardware and operating systems, file processing and database organization, data communication and networking, and, of course, programming. But what does such a program teach the student about working in industry? This, paper, written from the point of view of a training analyst for a major corporation who has had over ten years of experience teaching and developing courses and curriculum for a small college, suggests that many computer science graduates lack the ability to integrate the concepts "learned" into a skill set and a pattern of approaching problems that would enable them to fit easily into a business organization and to succeed in such an environment. The paper outlines types of skills necessary in business/industry and makes suggestions for integrating the development of these skills into the computer science curriculum. Over fifty graduates from undergraduate programs throughout the southeast were interviewed by the author. Approximately half of the graduates were already working in computer related positions; the others were in the process of interviewing for such positions.

The average graduate of a four-year program in computer science has spent approximately half of his time in computer related classes, learning about hardware, operating systems, networks, analysis and design of systems, and, of course, programming. Most graduates have a base of theoretical knowledge upon which practical skills can be built. They can write programs in two or three different languages, they know the difference between a CPU and disk drive; they can tell you the steps in analyzing and designing a system; and they can discuss the advantages and disadvantages of object oriented programming. But have they learned the skills that will help them to land a job and to survive in the workplace? What activities/requirements could be added to the courses already in place that would enhance their ability to succeed? The following is a discussion of the nontechnical, nontheoretical skills necessary to survive and advance in the work place. This discussion is not meant to be all inclusive, but merely a starting place for incorporating such skills into computer courses.

General skills required in the workplace:

1. SKILL: *Ability to manage his time and setting priorities.* In most cases, the graduate has learned how to manage his time, developing schedules for himself that include time for working, studying, and relaxing. He has learned to juggle several tasks at one time, such as studying for

a calculus exam, writing an English paper, completing a computer program, and finishing a project at work all in the same time period.

ACTIVITIES: Assign varying numbers of projects. Some of the projects should have the same due date, some different due dates. The projects should have varying levels of difficulty throughout the course - assigning increasingly difficult projects as the courses progresses gives a false impression of projects tackled in nonacademic situations. Within reason, "last minute" projects should occasionally be assigned.

2. SKILL: *Work effectively as part of a team,* understanding that the goals of the team often supercede the goals of the individual and that teams necessarily have both "stars" and "workers", are important concepts in the workplace. When working in a team, if one member does not pull his weight, the other team members must pick up the slack. However fair or unfair it may seem, a team oriented organization holds the entire team responsible for both success and failure. As far as management is usually concerned, late projects are the fault of the team. Cooperation - seeing the situation from other points of view, sharing resources, getting along with non computer people, and adapting to different management styles - is the basis of working effectively as part of a team.

   ACTIVITIES: As often as possible assign students to groups, making sure that each student has the chance to be a leader and a chance to be a "follower". Groups should vary in size, allowing the individual groups to work out sub groups, if necessary, as well as individual roles. Evaluation procedures should be developed to evaluate the group as a whole and the individuals within the group. Discussion of the group experience is a necessary follow-up activity . If individuals in a group are working well together adding a "ringer" who will not do his assigned tasks or who disrupts the groups is a valuable short term method for helping the groups learn to solve problems among its members.

3. SKILL: *Generalize learning to various situations.* A successful employee knows how to learn new concepts and how to apply old concepts to new situations.. He does not expect an instructor to explain everything to him. He has learned to dig through user manuals, help screens, and libraries.

   ACTIVITIES: After ensuring that the necessary resources are available, have the student modify a program written in a language he has not used yet. After a student has learn to use a specific operating system well, assign him projects to complete on other operating systems so that he will have to learn how systems are similar and different.

4. SKILL: *Managing stress.* The individual needs to be able to manage stress and stressful situations in such a way that the situation does not affect his health or his ability to complete the task at hand. The stress may be due to continually changing deadlines and specifications for jobs or due to interactions with a difficult co-worker, boss, or customers. The individual must learn to be flexible in his expectations, adjusting to changes in plans, deadlines.

   ACTIVITIES: Discussions and role playing are effective methods for helping students deal with stress. The discussions should include how to separate work and personal time, how to deal with

last-minute demands, how to deal with difficult people in order to create a win-win situation, how to take mini-breaks, and when to walk away from a situation.

5. SKILL: *Communicate effectively.* The individual should be able to use verbal and written communication well. He should be able to present his ideas in a clear coherent manner. The individual should be able to interact and communicate successfully with various types of people, asking questions, seeking information.

   ACTIVITIES: Include frequent mini papers to allow the students to practice written communication in their field (papers in freshman English classes are not the same concept). Hold both impromptu and planned discussions and debates in classes covering topics of interest to the field. Have student do frequent informal and formal presentations in front of small and large groups. The more practice students have with communication skills, the more comfortable the students will be with communicating effectively.

6. SKILL: *Supporting ideas and problem solutions.* Not only must students be able to form solutions to problems, but they also must be able to explain why their solution is reasonable. They must be able to understand their ideas clearly enough to be able to defend them to themselves and to others.

   ACTIVITIES: From the first course, students should be asked to present their solutions to small and large groups, justifying their approach to the problem and suggesting alternative methods of solution. Debates, both formal and informal, on relevant issues in computer science are effective tools for crystallizing ideas and concepts.

"Computer" skills required in the workplace:

1. SKILL: *Logical thinking* as illustrated through the development of algorithms. Algorithms are evident in the solutions of all problems, whether writing computer programs, designing a system of some sort, or planning a route to a new destination.

   ACTIVITIES: Before taking any programming course, students should be introduced to the design solutions to problems using words, diagrams, and a combination of both. A traditional example is to have a team of students explain how to tie a shoe, giving the procedure to a second team to actually follow. Additional problems, such as designing a queuing system at the local fast food place, or setting up the schedule for a softball league, bring practical situations into very discreet steps. This introduces the concepts of logical design, walkthrus, and modification before any programming language is used.

2. SKILL: *Eliminating program errors.* In the workplace, bugs are errors and they are not acceptable - a program must work for all data, all the time. Programming errors can increase the cost of producing a product by causing program re-runs, product recalls, and lost business.

   ACTIVITIES: From the very first course, only complete, correct, and error-free programs and assignments should be accepted from students. Using the term "error" instead of "bug" to refer to mistakes in software emphasizes the fact software problems are unacceptable. Thus, from

the very first project, students should perform walkthrus on their code and on the code of other students. When testing a program, students should develop test data to cover all possible situations. Documentation of the systematic testing of the programs should be kept. Incomplete programs and incompletely tested programs should not be accepted - there is no partial credit in the real world!

3. SKILL: *Reusing software* is not only a timesaver but it is also critical to the high productively levels expected from programmers - as far back as subroutine and functions in FORTRAN, code has efficiently been reused in order to save testing and coding time. Starting each program from scratch may be personally satisfying for a student, but it is not how the world operates. Reduction of errors, testing time, and programming time are key elements in increasing programmer productivity.

   ACTIVITIES: Introduce student early to the concept of building their own libraries by using modules they have coded and thoroughly tested as the basis of their library. Using modules from classmates (after proper permission is obtained and proper credit to the original author is included in the module) and from other sources, saves time when the modules have been well-documented and thoroughly tested.

4. SKILL: *Adjusting to constantly changing project specifications*. Students are used to getting a programming assignment, coding it, and turning it in with no changes in assignment specifications. But customers do change their minds, frequently. Students need to be able to make modifications in their code easily and efficiently - a well documented and modulized program makes this possible.

   ACTIVITIES Hand out a programming assignment with a specified due date. On the date due, have the students make "last minute" changes to the program before turning it in. Allowing time in class helps students adjust to the stress of such a change without rewarding those who wait for the last minute to code their projects. Gradually increase the difficulty of the projects and the complexity of the modifications.

5. SKILL: *Developing good design skills*. The cost of trial and error programming is not readily evident to most students. They rarely have to pay for CPU time and they rarely see a clock ticking off CPU time, especially in the PC oriented labs at most schools. As long as an assignment is completed they usually can not tell exactly how long the assignment took to code and test.

   ACTIVITIES: Limiting students to a set amount of computer time or lab time on a particular project helps make them aware of the type of planning that should take place before sitting down to code at a machine. Using a manageable project, have students fill out a "timesheet" that includes spaces for designing the program, writing the algorithm, writing the code, coding the project, correcting syntax errors, correcting logic errors. Done over a series of projects, such a timesheet should assist students in observing the negative correlation between the time spent designing and writing the algorithm and the time spent debugging logic errors.

In conclusion, although students need to learn a specific body of knowledge to be considered computer scientists, simply teaching them theories and information will not allow them to succeed in the field. They must learn how to effectively apply the knowledge in the real world. They must also be able to communicate ideas effectively, work with a variety of people, and expand their knowledge and skills as the field changes.

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement (OERI)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐

This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").