Should Computer Programming Be Taught To All First-Year Engineering Students?

Byron S. Gottfried
University of Pittsburgh

Abstract

This paper presents the pros and cons of the programming requirement for all first-year engineering students, based upon the results of a survey of the engineering faculty at the University of Pittsburgh. The results show that the computing skills required of most frequently required of. engineering students, in decreasing order, are:

1 Basic computer skills

2 Word processing

3 Spreadsheets

4 Equation solvers and programming languages

A rational alternative is suggested, in which *all* students would be taught basic computing skills, technical uses of spreadsheets, and equation solvers during their first year. *Some* students would then receive instruction in a programming language at a later time, based upon need or interest.

Introduction

For the past three decades, most colleges and universities have required their first-year engineering students to take a course in computer programming. The reasoning behind this originated in the 1960s, when it was widely believed that engineers must know how to program in order to solve engineering analysis problems on a computer.

Since the 1960s, the world of computing has changed dramatically. Mainframe computers were replaced by minicomputers, which were later replaced by workstations and personal computers. Batch processing gave way to timesharing, and then to interactive computing using graphical user interfaces on personal computers and workstations. Computer prices fell by three orders of magnitude, while computers became faster and more reliable. Inexpensive software, some of excellent quality, became widely available for many different purposes, including engineering applications. Today most practicing engineers, and a growing number of engineering students, own their own computers and several supporting software packages.

Today's engineers no longer need to write customized computer programs in order to solve many of the problems that commonly arise in engineering analysis. However, many engineering educators have ignored this paradigm shift, and still believe that a course in computer programming should be required of all first-year engineering students. A growing number of engineering educators is now beginning to question this 35-year-old idea, however, suggesting instead that first-year students be taught to use commercial software packages to solve engineering problems, and that programming be taught either as an elective, or as a requirement for those students whose disciplinary interests require it (such as computer engineers).

This paper reports the results of a survey of the full-time faculty within the School of Engineering at the University of Pittsburgh on required student computing skills. The survey asked detailed questions in each of the following two general areas:

1. What *prerequisite* computing skills are required of students entering each professor's class, and at what level of proficiency? _
2. What *new* computing skills are taught within each class, and at what level of proficiency?

The questions go beyond the issue of required programming languages, in order to broaden the information base and enhance objectivity.

The results raise some interesting questions and indicate the need to probe further into certain of the faculty responses.

The Survey Questions

Faculty members were asked to answer the survey questions for each class that they teach. Each question was to be answered by providing an integer value between O and 4, where O is a low response (no skill required or taught) and 4 is a high response (high level of proficiency required or taught).

I. What prerequisite computer skills do you require of your incoming students for each of your courses?

1. Basic computer skills (viewing a directory, copying files, etc.)
2. Word Processing
3. Spreadsheet applications (including simple data analysis, graphing)
4. Equation solvers (Mathematical, MathCad, etc.)
5. Specialized data analysis, including statistical packages
6. Drawing/Graphical modeling (AutoCad, Pro Engineer, etc.)
7. Advanced analysis packages (FEA, SPICE, etc.)
8. Simulators and special-purpose simulation languages
9. General-purpose programming languages (Fortran, C, Pascal, etc.)
10. Other (please specify)

II. What computer skills do you teach in each of your courses?

1. Basic computer skills (viewing a directory, copying files, etc.)
2. Spreadsheet applications (including simple data analysis, graphing)
3. Equation solvers (Mathematical, MathCad, etc.)
4. Specialized data analysis, including statistical packages
5. Drawing/Graphical modeling (AutoCad, Pro Engineer, etc.)
6. Advanced analysis packages (FEA, SPICE, etc.)
7. Simulators and special-purpose simulation languages
8. General-purpose programming languages (Fortran, C, Pascal, etc.)
9. Other (please specify)

Responses

The survey was sent to about 80 faculty members (the exact number is unclear, since they were placed in faculty mail boxes on a departmental basis). Thirty-two faculty members responded. The table shown below summarizes the responses, showing the number of responses at each level (O through 4) for each question. The

total number of responses and the weighted average score are also shown for each question. (Note that most faculty responded for two or more courses. Hence, the totals shown for some questions exceed 32).

There were 299 responses for all questions in Category I, and 92 responses for all questions in Category II.

In Category I (Required Prerequisites), the number of responses for each question ranged from 71 to 6 (excluding the "other" question), and the weighted averages ranged from 2.44 to 1.33. The overall weighted average (the response-weighted average of the individual averages) for Category I was 2.17.

In Category II the number of responses for each question were lower, ranging from 15 to 3 (excluding the "other" question). The weighted averages were higher, however, ranging from 3.33 to 1.6, though the overall weighted average (the response-weighted average of the individual averages) for Category II was only 1.85 (because of the much larger number of low-valued responses).

The listings of "other" topics is quite broad, reflecting various disciplinary interests.

I. Required Prerequisites

1- Basic computer skills

0	1	2	3	4	Total	Average
	20	22	17	12	71	2.30

2- Word processing

0	1	2	3	4	Total	Average
	19	14	16	12	61	2.34

3- Spreadsheet applications

0	1	2	3	4	Total	Average
	17	10	12	3	42	2.02

4- Equation solvers

0	1	2	3	4	Total	Average
	9	14	8	1	32	2.03

5- Specialized data analysis

0	1	2	3	4	Total	Average
	5	14	2		21	1.86

6- Drawing/ graphical modeling

0	1	2	3	4	Total	Average
	2	12	7		21	2.24

7- Advanced analysis packages

0	1	2	3	4	Total	Average
	4	2			6	1.33

8- Simulators, simulation languages

0	1	2	3	4	Total	Average
	2	4	1	2	9	2.33

9- General-purpose programming languages

0	1	2	3	4	Total	Average
	4	14	10	4	32	2.44

10 - Other - . ..2 -

0	1		3	4	*Total*	*Average*
	1	1	1	1	**4**	**2.50**

Specific topics mentioned misc. canned packages, linear programming, transportation, intemet, process simulators, Unix, X-Windows, Matlab, ladder logic (programmable logic controllers)

II. New Topics Taught in Class

1- Basic computer skills

0	*1*	2	3	4	*Total*	*Average*
	8	**6**	0	**1**	*15*	*1.60*

2- Spreadsheet applications

0	*1*	2	3	4	*Total*	*Average*
	7	**4**	**3**		**14**	*1.71*

3- Equation solvers

0	*1*	2	3	4	*Total*	*Average*
	6	**2**	**3**		11	*1.73*

4- Specialized data analysis

0	*1*	2	3	4	*Total*	*Average*
	4	**2**	**1**	**2**	*9*	*2.11*

5- Drawing/ graphical modeling

0	*1*	2	3	4	*Total*	*Average*
		1		**2**	*3*	*3.33*

6- Advanced analysis packages

0	1	2	3	4	*Total*	*Average*
		3		**2**	*5*	*2.80*

7- Simulators, simulation languages

0	1	2	3	4	*Total*	*Average*
		2	**3**	**5**	10	*3.30*

8- General-purpose programming languages

0	1	2	3	4	*Total*	*Average*
	9	**3**	**0**	**3**	**15**	*1.80*

9- Other

0	*1*	**2**	**3**	**4**	*Total*	*Average*
		7	*3*		10	*2.30*

Specific topics mentioned intemet, presentation software, process simulators, Unix, X-Windows, CAD, Matlab, ladder logic (programmable logic controllers), NC Language, Hypercard, discrete-event simulation language, Adams

Discussion of Results

Some interesting inferences can be drawn from these results. First, it is clear that most faculty who require computing skills of their students expect those skills to be taught elsewhere. This can be seen from the 299 Category I responses and only 92 Category II responses.

Second, of those faculty members who do teach computing skills within their own courses, most provide a slightly lower level of proficiency than the prerequisite proficiencies expected of incoming students. (Compare the overall Category II average of 1.85 with the overall Category I average of 2. 17). The sub-categories of Drawing/Graphical Modeling, Advanced Analysis Packages, and Simulators/Simulation Languages are notable exceptions, though the first two sub-categories elicited very few responses (3 and 5 responses, respectively).

Third, the Category I responses show little variation in the individual averages (other than the small Advanced Analysis sub-category), but large variations in the number of responses. The sub-category of Basic Computer Skills shows the largest number of responses (7 1), followed by Word Processing (61) and Spreadsheet Applications (42). Thus, the required prerequisite skills that are cited most frequently are basic computer skills, word processing and spreadsheet use. This is not at all surprising.

Of particular interest in this study is the issue of general-purpose programming languages. At face value, it appears that a fair number of faculty members require programming skills as prerequisites. However, this sub-category received a total of only 32 responses, causing it to rank fourth (tied with equation solvers) in terms of number of responses. The Spreadsheet and Equation Solver sub-categories, which provide many of the same problem-solving tools, collectively received 42 + 32 = 74 responses – more than double. This is consistent with the recent historical trend toward the use of problem-solving tools rather than programming languages for solving applied problems in engineering courses.

Moreover, of these 32 responses indicating a need for programming skills, it is reasonable to hypothesize that *some* faculty who require programming skills of their students do so only because that is what they themselves know. For some applications, their students could use a spreadsheet program or an equation solver to solve the same problems with less effort.

Finally, consider the Category II question regarding instruction in computer programming within a course. Of the 15 positive responses, three relate to the introductory freshman programming course, and most others come from two departments that favor Fortran rather than C (the language taught to all freshmen). Thus, it appears that these responses largely reflect a desire for a second (redundant) language rather than additional skills in the primary language. Again, it is likely that some faculty are requiring students to replicate their own knowledge base acquired some years ago.

Conclusions

The following conclusions can be drawn from this study, based upon a survey of the engineering faculty at the University of Pittsburgh:

1. Engineering faculty generally expect their students to have prerequisite computer skills before entering class.
2. Skills in word processing and spreadsheet use are considered more important than programming skills by most faculty. The use of problem solvers is considered equally important to programming skills.
3. There is a legitimate demand for some students to acquire programming skills, in areas such as computer engineering. In other cases, however, faculty members may require their students to write programs because that is what they (the faculty) know.
4. An alternative to the traditional first-year programming course might be a course in basic computer skills, spreadsheet use and the use of an equation solver. Programming could then be taught in greater depth to those students who need it (e.g., computer engineers) or who may wish to take it as an elective.

5. Faculty members should be strongly encouraged to update their computing skills periodically, so that they know the best tools for various disciplinary-oriented technical applications,

BYRON S. GOTTFRIED

Byron S. Gottfried is a Professor of Industrial Engineering and Academic Director of the Freshman Engineering Program at the University of Pittsburgh. His interests include computer simulation and software engineering as well as new educational paradigms and the development of educational software. Dr. Gottfried also taught at Carnegie-Mellon University, and was formerly employed by NASA, Gulf Oil Corporation and Westinghouse Electric. He is the author of several textbooks on computer programming and computer applications in engineering.