

# What is a Fully Qualified Engineer?

R.A. Wattenbarger, SPE, Texas A&M U.

In this article, I focus on the computer-related skills that an experienced (5-year) engineer should have to be fully effective. What should an engineer know and be able to do with the computer and how does he/she learn these skills?

This topic comprised a discussion session during last year's SPE Forum Series on computers. Like most of the topics, it spurred considerable discussion and differences in opinion, but there were also some common thoughts.

During the discussion, we recognized that none of us knew exactly what the computing environment would be like in our target time of 5 years. We certainly can look back and see that some of our forecasts about the future simply did not materialize, while other unexpected events did. There was a time when the ALGOL language was thought to be the superior programming language and would eventually take over. Instead, ALGOL eventually died (many of its features were adopted by other languages). Companies dedicated to ALGOL libraries and compatibility had a hard time deciding when and how to change over. More recently, many expected Unix to become the universal standard. Computer experts wanted a single standard that would apply to the smallest PC and the largest mainframe computer. Instead, we see a continued prominence of the DOS/Windows environment—driven by the market, not by the high-level planners of the computer industry. (This reminds me of a joke that was popular when I was in college: "no one likes Lawrence Welk but the public.") We have a hard time controlling and predicting the future, but we must continue to try.

My perspective comes from a computing career that covers more than 30 years as an engineer, a software developer, and an educator. I wrote my first FORTRAN program, a separator-optimization program, in 1963. Although the computing facilities and style were very clumsy then, I found learning a computer language fascinating. During this programming task, I was forced to learn more about the details of phase-equilibrium calculations than I had learned in school or would use as a practicing engineer. This programming experience then led to a full-time assignment in a group developing applications software for economic evaluation of mergers. During that task, I learned more details about taxes, accounting, and economic evaluation than other practicing engineers.

Maybe there is a thread of commonality in this ancient experience that applies today, or, more importantly, will apply 5 years from now.

## What Computer Skills Will We Need?

How do we know what computer skills will be required? Recently, some surveys have been taken to find out.

**Louisiana State U. (LSU) Survey.** The faculty of LSU surveyed their past graduates to determine what computer skills were the most important to them in their jobs. The survey indicated that the most useful computer applications involved spreadsheets, word processors, etc.—in other words, the equivalent of MS Office. FORTRAN programming fell at the bottom of the list. We would expect that the majority of these engineers are BS-level graduates working in practically oriented jobs. Software development is obviously not part of their jobs.

**Stanford Survey.** Khalid Aziz, of Stanford U., surveyed his current graduate students to find what computer skills they considered to be the most important. FORTRAN topped the list, closely followed by C++. Also high on the list were statistical and mathematical application programs.

The results of the Stanford survey appear opposite of the LSU survey. Why? Who knows? However, some differences are apparent,

such as current students (Stanford) vs. practicing engineers (LSU), graduate students vs. BS graduates, and a more research-oriented program vs. a more practical program. There are probably other differences, but the point is that it is not obvious what computer skills should be emphasized.

We expect an engineer to be "good with" computers—both PC's and Unix workstations. Here is my list of the computer skills I think an engineer will be expected to have in the future.

1. Good working knowledge of PC hardware and operating software.
2. Working knowledge with Unix in a workstation environment.
3. An ability to work easily in a Windows environment.
4. An ability to install applications software on a PC or Unix computer.
5. Working knowledge of a source programming language.
6. Active e-mail familiarity and usage.
7. Access to and ability to use Internet to access electronic publishing and data sources.
8. Working knowledge of large database standards and database software.

## How Do We Prepare?

The ideal way to learn about computers and computer applications is to be a professor and learn from students. We are surrounded by students who know details about application packages that are difficult to learn from documentation and tutorials. Sometimes we make class homework assignments, then ask students how they solved the problem and learn from them. They can become very proficient in certain topics, particularly during their research projects. They always find the time to sit down with someone to transfer their knowledge.

But this student resource is not available in industry. I am sure that a great deal of learning still comes from fellow workers, but these fellow workers may be less willing to sit down and explain or demonstrate computer knowledge. Engineers are more likely to be focused on their own engineering priorities.

**The Role of Universities.** Universities cannot do everything. There are certain things that we do well and other things that we do not do well. There are some areas that we should avoid. Companies cannot teach calculus well, and universities do not teach field experience well. Although we often try to enliven our lectures and problem assignments by relating them to actual field problems, I am always amazed at how fast a new graduate starts talking about practical field problems after only a few months on the job.

Course work in universities can have a wide range in purpose, objectives, and content. An age-old controversy arises about the objectives of a particular course once the topic has been selected. The general tendency of professors is to teach derivations, concepts, and emphasize fundamentals of a particular subject. This is simply "the academic process." Students consider much of this boring, irrelevant, and tedious. Students are much more interested in just learning the formulas and hearing about actual field applications and examples of what a practicing engineer actually does—the more recent and closer to home, the better. Some call the difference in these viewpoints education vs. training.

It is obvious to me that we should concentrate on teaching the building blocks of a technical career instead of trying to jump to the latest technology. The student has a better education if he/she learns the principles involved in a calculation rather than learning the latest calculation. So, the question is, what are the important fundamentals in computing that will allow the engineer to grow with changing technology?

Learning to program in a source language, such as FORTRAN, is a must. Although students would much rather learn a new "point and click" program or use a spreadsheet for making a calculation, there is a certain training process involved in learning the step-by-step process of programming logic and structure. The mind training is vital to become proficient in using engineering applications, especially when something goes wrong.

**Observations and Assertions About Computer Skills.** The following illustrate some of my thoughts on learning/applications levels.

1. A student who learns FORTRAN in school will learn spreadsheets on the job. A student who learns spreadsheets in school will not learn FORTRAN on the job. The student who learns FORTRAN in school will eventually be better at both FORTRAN and spreadsheets.

2. A student who learns DOS in school will learn Windows easily. A student who learns only Windows in school will probably not learn DOS on the job. The student who learns DOS in school will eventually be better at both DOS and Windows.

3. A student who has learned a first language, say FORTRAN, in class can learn a second language, say C++, easier than a student who has not learned a first language. The process of learning a programming structure, learning the syntax, and having some experience in the language transfers to the second language. This is not unlike spoken languages. People who have become fluent in a second language usually have less trouble learning the third language (this is complicated by the age when one learns the languages, of course).

These observations would lead to the conviction that learning programming and basic computer commands is a must for formal education. I and many of my colleagues take this viewpoint. However, there are others who believe that computer usage should be taught on a modern usage basis only. They would contend that learning DOS and FORTRAN (or another programming language) is slow, tedious, obsolete, and boring to students. So, we are presented with the balance of teaching basic thinking processes and keeping up with the modern world of computing.

**Continuing Education.** There is little continuing education in computer skills from universities. I do not see this as the proper role of universities. We should stick to more fundamental types of education. The best sources of continuing education are probably seminars from vendors, internal seminars, and informal exchanges with fellow workers.

**The Age Factor.** It is no surprise that younger engineers learn computer usage faster than older engineers. There are several factors that contribute to this: (1) as we grow older we probably do not have the same learning capacity, (2) we tend to rely on our experience more and sometimes this includes older computing methods, and (3) we tend to spend more of our time managing, meeting, and mentor-

ing and not as much time actually using computers. We probably also have less of a tolerance for long computing sessions. I marvel at the ability of students to spend hours in front of a computer screen. During our "maturing" process, it becomes important to rely on others for computing expertise and advice. But one must guard against becoming so dependent on the expertise of others that one loses fundamental competence and finds himself/herself "out of the loop."

### **The Key—Engineering Principles**

One theme cuts through all the discussion about details of computing and teaching: the most important elements of engineering training and skills are the principles of engineering itself.

The modern version of the "cookbook" engineer is one who uses computer applications blindly without understanding the principles involved. This can be dangerous in many ways. Bugs in the program may go undetected because of a desire to go directly from the data to the answer. Often, it can be crucial to understand the underlying assumptions of a method or how a method compares to alternate methods and how to select the best method.

For years, I have worked with inexperienced users in reservoir simulation. There is a great tendency to say "just tell me what data I need and how to input it." This type of usage can often lead to improper description of the problem, inefficient use of the simulator, or just plain wrong answers. A user should have a good understanding of the computational and physical principles and take steps toward learning the technology or, at least, get advice from experts.

### **Conclusion**

An engineer will have continually to invest at least a small fraction of his/her time in keeping up with computer trends and applications and staying proficient. This will be a prerequisite. A successful engineering career will depend much more on understanding the principles of science and engineering and keeping up with oil industry technology.

---

**Robert A. Wattenbarger** is a professor of petroleum engineering at Texas A&M U. in College Station. He was previously a vice president and director of Scientific Software-Intercomp Inc. and has worked in reservoir simulation and software development for years. He holds BS and MS degrees from The U. of Tulsa and a PhD degree from Stanford U., all in petroleum engineering.

---

**SPEAKER'S CORNER** is a forum available for discussion of computer issues that relate to the petroleum industry. If you have a selection you'd like us to consider, please send it to [vdawe@spelink.spe.org](mailto:vdawe@spelink.spe.org).