Defining "Engineer:" How To Do It and Why It Matters

MICHAEL DAVIS
Center for the Study of Ethics in the Professions
Illinois Institute of Technology

ABSTRACT

Controversy concerning whether "software engineers" are, or should be, engineers provides an opportunity to think about how to define "engineer" and what effect different definitions may have on our understanding of engineering. The standard definitions of engineering are shown to generate more confusion than insight. Engineering should be defined historically, as an occupation, and ethically, as a profession. An engineer is a member of the engineering profession, that is, a member both of an occupation that is engineering by "birth," "adoption," or "marriage" and of the profession committed to engineering's code of ethics. Today, few "software engineers" satisfy either of these conditions. It is an open question whether they should.

I. INTRODUCTION

"Today, the field [of software engineering] has emerged as a true engineering discipline."

—*John J. Marciniak, "Preface," Encyclopedia of Software Engineering,* 1994

"If you are a 'software engineer,' you could be breaking the law. It is illegal in 45 states to use that title, warns *Computerworld* newspaper. People who aren't educated and licensed in 36 recognized engineering disciplines can't call themselves 'engineers,' and computer professionals often don't qualify."

—*Wall Street Journal, June 7, 1994, p. 1.*

This paper begins with recent events in what may or may not be the history of engineering, the emergence of "software engineering" as a distinct discipline. The term "software engineering" seems to have come into currency after a NATO conference on the subject in 1967.[1] Today, many thousands of people are called "software engineers," do something called "software engineering," and have sophisticated employers willing pay them to do it.*

Yet, "software engineering" is no ordinary engineering discipline. Many "software engineers" are graduates of a program in computer science having a single course in "software engineering." Typically, that course is taught by someone with a degree in computer science rather than engineering. While there seem to be no hard numbers, most "software engineers" are probably programmers with no formal training in engineering. Are "software engineers" nonetheless engineers?

* "In the 1991 Computer Society [of the Institute for Electrical and Electronic Engineers] membership survey, over half (54 percent) of the current full members polled indicated that they consider themselves software engineers, as did 40 percent of the affiliate members."[2]

II. THE STANDARD DEFINITION OF "ENGINEER"

The standard definition of "engineer" looks something like this: An engineer is a person having at least one of the following qualifications: a) a college or university B.S. from an accredited engineering program or an advanced degree from such a program; b) membership in a recognized engineering society at a professional level; c) registration or licensure as an engineer by a government agency; or d) current or recent employment in a job classification requiring engineering work at a professional level. (This definition, the work of the National Research Council's Committee on the Education and Utilization of the Engineer, appears in reference 3.) The striking feature of this definition is that it *presupposes* an understanding of the term "engineering." Three of the four alternatives actually use the term "engineering" to define "engineer;" and the other, alternative c), avoids doing the same only by using "as an engineer" instead of "to practice engineering."*

This definition, and those like it, are important. They determine who is eligible for admission to engineering's professional societies, who may be licensed to practice engineering, and who may hold certain jobs. Such definitions are also eminently practical. For example, they do in fact help the Census Bureau exclude from the category of engineer drivers of railway engines, janitors who tend boilers in apartment buildings, and soldiers wielding shovels in the Army's Corps of Engineers. These, though still called "engineer," clearly are not engineers in the relevant sense.

The standard definitions do not, however, suit our purpose. They will not tell us whether a "software engineer" is an engineer—or even how to go about finding out. A "software engineer" may, for example, work at a job classified as requiring "software engineering [at a professional level]". That will not settle whether she is an engineer: what an employer classifies as "engineering" (for lack of a better word) may or may not be engineering (in the relevant sense).**

What will settle the question? In practice, the decision of engi-

*Compare the more elegant Canadian definition[4]: "The 'practice of professional engineering' means any act of planning, designing, composing, evaluating, advising, reporting, directing or supervising, or managing any of the foregoing that requires the application of *engineering* principles, and that concerns the safeguarding of life, health, property, economic interests, the public welfare or the environment." (Italics mine.) The report contains no definition of "engineering principles."

**Similar problems arise for "genetic engineer" and might arise for other "engineers," for example, "social engineers." (This problem of definition is, of course, not limited to engineers: lawyers are no more successful defining "the practice of law," or doctors "the practice of medicine.")

neers. An organization of engineers accredits baccalaureate and advanced programs in "engineering." Other organizations of engineers determine which societies with "engineer" in the title are "engineering societies" and which—like the Brotherhood of Railway Engineers—are not. Engineers also determine which members of their societies practice engineering ("at a professional level") and which do not. Government agencies overseeing registration or licensure of engineers, though technically arms of the state rather than of engineering, generally consist entirely of engineers. And, even when they do not, they generally apply standards (education, experience, proficiency, and so on) that engineers have developed. Engineers even determine which job classifications require engineering work ("at a professional level") and which do not.

The decision of engineers has settled many practical questions, but not ours. Engineers divide concerning whether "software engineering" is engineering (in the appropriate sense).[5] So, to say that "software engineers" are engineers if, in the opinion of engineers, they engage in engineering (at the professional level) is—for engineers and those who rely on their judgment in such matters—merely to restate the question.*

That is a practical objection. There is a related theoretical objection. A definition of "engineer" that amounts to "an engineer is anyone who does what engineers count as engineering" violates the first rule of definition: "Never use in a definition the term being defined." That rule rests on an important insight. Though a circular definition can be useful for some purposes, it generally carries much less information than a non-circular definition; it generally conceals foundational questions rather than helping to answer them.

How might we avoid the standard definition's circularity? The obvious way is to define "engineering" without reference to "engineer" and then define "engineer" in terms of "engineering." The National Research Council (NRC) in fact tried that approach, coupling its definition of "engineer" with this definition of "engineering:" "Business, government, academic, or individual efforts in which knowledge of mathematics and/or natural science is employed in research, development, design, manufacturing, systems engineering, or technical operations with the objective of creating and/or delivering systems, products, processes, and/or services of a technical nature and content intended for use."

This definition is informative insofar as it suggests the wide range of activities which today constitute engineering. It is nonetheless a dangerous jumble. Like the standard definition of "engineer," it is circular: "Systems *engineering*" should not appear in a definition of "engineering." The same is true of "technical" if used as a synonym for "engineering." (If not a synonym, "technical" is even more in need of definition than "engineering" is and should be avoided for that reason.) The NRC's definition also substitutes uncertain lists—note the "and\or"—where there should be analysis. Worst of all, the definition is fatally overly inclusive. While "software engineers" are engineers according to the definition, so are many whom no one supposes to be engineers. Not only do applied chemists, architects, and patent attorneys *clearly* satisfy the definition, but—thanks to the "and/or" between "mathematics" and "natural science"—arguably so do actuaries, accountants, and others who use mathematics to create financial instruments, tracking systems, and other technical objects for use.

Though much too inclusive, this definition of "engineering" shares with most others three characteristic elements: First, it makes mathematics and natural science central to what engineers do. Second, it emphasizes physical objects or physical systems. Whatever engineering is, its principal concern is the physical world *rather than* rules (as in law), money (as in accounting), or even people (as in management). Third, the definition makes it clear that, unlike science, engineering does not seek to understand the world but to remake it.* Engineers do, of course, produce knowledge (for example, tables of tolerances or equations describing complex physical processes), but such knowledge is merely a means to making something useful. Engineers also produce beautiful objects (for example, the Brooklyn Bridge or the typical computer's circuit board). They are nonetheless not artists (in the way architects are). For engineering, beauty is not a major factor in evaluating work; utility is.

Those three elements, though characteristic of engineering, do not define it. If they did, deciding whether "software engineers" are engineers would be far easier than it has proved to be: we could show that "software engineers" are not engineers simply by showing that they generally do not use the "natural sciences" in their work. That many people, including some engineers, believe "software engineers" to be engineers is comprehensible only on the assumption that these three characteristics do not define engineering (except in some rough way). But if they do not define engineering, what does?

Before answering that question, I shall describe three common mistakes about engineering to be avoided in any answer.

III. Three Mistakes about Engineering

The NRC's definition of "engineering" uses "technical" twice, once as a catch-all ("or *technical* operations") and once to limit the domain of engineering ("of a *technical* nature and content"). It is the second use of "technical" that concerns us now. It seems to be an instance of a common mistake about engineering, one even engineers make. We might summarize it this way: *engineering equals technology.* Anyone who makes this mistake will find it obvious that engineers, whatever they may have been called, have been around since the first technology (whether that was the Neanderthal's club, a small irrigation system, or a large public building). Engineering will seem the second oldest profession.

There are at least three objections to this way of understanding engineering. First, engineering can equal technology only if we so dilute what we mean by "engineering" that any tinkerer would be an engineer (or, at least, be someone engaged in engineering). Once we have so diluted engineering, we are left to wonder why anyone would want a "software engineer" rather than a "programmer," "software designer," or the like to do software design or development? What was the point of inventing the term "software *engineering*?" (Note, for example, reference 6: "[The] term software en-

*The IEEE has defined "software engineering" as "application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software: that is, the application of engineering to software."[2] This definition (or, rather, that "that is") begs the question whether the systematic, disciplined, and quantifiable approach in question is an application of engineering to software or the application of a different discipline. Not all systematic, disciplined, and quantifiable approaches to development, operation, and maintenance are necessarily engineering. Indeed, the fact that software is primarily not a physical but a mathematical (or linguistic) system certainly suggests that engineering principles have, at best, only limited application.

*Compare the Smetonian Society's now classic definition of "civil engineering": "...the art of directing the great sources of power in Nature for the use and convenience of man."

gineering expresses the continued effort to put programming into the ranks of other engineering disciplines.")

Engineering societies adopt standards to help society know what to expect of engineers, not to tell a conscientious and technically adept [illegible]

The *corps du génie* soon began realizing this potential. Within two decades, they were known all over Europe for unusual achievements in military construction. When another country borrowed the French word "engineering" for use in its own army, it was for the sort of activity the corps du génie engaged in.[9,10] That was something for which other European languages lacked a word.

But, as of 1700, the *corps du génie* was not a school of engineering in our sense; it was more like an organization of masters and apprentices ("proto-engineers" of various degrees). I speak here not of the craftsmen in the corps but of the officers. Though engineers sometimes describe themselves as deriving from craftsmen of one kind or another, the way I am telling their story, they derive from military *of-*

were geometry, trigonometry, physics, and the fundamentals of chemistry with practical applications in structural and mechanical engineering. There was a good deal of drawing, some laboratory and workshop, and recitations after each lecture. The second and third year continued the same subjects, with increasingly more application to the building of roads, canals, and fortifications and the making of munitions. For their last year, students were sent to one of the special schools: the school of artillery, the school of military engineering, the school of mines, the school of bridges and roads, and so on.[12]

Engineers will immediately recognize this curriculum, especially the four years, the progression from theory (or analysis) to application (or design), and the heavy emphasis on mathematics, engineering as a profession. The history of a profession tells how a certain occupation organized itself to hold its members to standards beyond what law, market, and morality would otherwise demand. The history of a profession is the history of organizations, standards of competence, and standards of conduct. For engineering in the United States, that history hardly reaches back to the Civil War.* It is a confused history because the profession was taking shape along with the occupation. Many early members of its professional societies would not qualify for membership today.

Nonetheless, we can, I think, see that, as engineers became clearer about what engineers were (or, at least, should be), they tended to shift from granting membership in their associations ("at a professional level") based on connection with technical projects, practical invention, or other technical achievements to granting membership based on two more demanding requirements. One—specific knowledge (whatever its connection with what engineers actually do)—is occupational. This requirement is now typically identified with a degree in engineering. The other requirement—commitment to use that knowledge in certain ways (that is, according to engineering's code of ethics)—is professional. While many professions (law and medicine, especially) make a commitment to the profession's code of ethics a formal requirement for admission, engineering has not (except for licensed P.E.'s). Instead, the expectation of commitment reveals itself when an engineer is found to have violated the code of ethics. The defense, "I'm an engineer but I didn't promise to follow the code and therefore did nothing wrong", is never accepted. The profession answers, "You committed yourself to the code when you claimed to be an engineer."[17]

Attempts to understand "software engineering" as engineering

question is, rather, whether (or when) they *should* be. There is no fact of the matter when identifying engineering disciplines, only a complex of social decisions in need of attention—especially, decisions about how to train "software engineers."

Acknowledgements

I read an early version of this article at the Center for Human Values, Princeton University, April 1, 1994. I should like to thank those present, as well as Ilene Burnstein, Deborah Johnson, Michael Loui, Michael Mahoney, Helen Nissenbaum, Vivian Weil, and the anonymous reviewers of this journal, for many of the improvements since then.

References

1. Gary A. Ford and James E. Tomayko, "Education and Curricula in Software Engineering," *Encyclopedia of Software Engineering*, v. I, John Wiley & Sons: New York, 1994, p. 439.

2. Fletcher J. Buckley, "Defining software engineering," *Computer*, vol. 26, no. Aug, 1993, p. 77.

3. Samuel Florman, *The Civilized Engineer*, St. Martin Press: New York, 1987, pp. 64-66.

4. Canadian Engineering Qualifications Board, *1993 Annual Report*, Canadian Council of Professional Engineers: Ottawa, 1993, p. 17.

5. "A Debate on Teaching Computing Science," *Communications of the ACM*, vol. 32, no. 12, 1989, pp. 1397-1414, especially those contributors