

# Software Engineering: A Profession in Waiting

**David Lorge Parnas,** Middle Road Software, Inc.

*Ongoing efforts to make software development an engineering discipline will fail until we have legislation requiring that creators of certain types of software be licensed, establishing a licensing authority, and detailing the capabilities that a licensed developer must possess.*

For more than 50 years, people concerned about software development have tried to make it a profession like medicine or civil engineering.[1] Those who started discussing "software engineering" in the 1960s had been trained in other fields; their work required computation and that led them to writing programs—often for their own use. A few were writing programs that would be used by others; such programs became known as *software*. Some of them observed that they were doing something very different from what they had been taught to do. They had been educated as mathematicians or scientists. Software development was more like engineering. They had been trained to extend knowledge but were now applying that knowledge to build products. That thought suggested the term *software engineering*. Those who introduced that term hoped that properly educated software developers would produce trustworthy products and thereby earn the good reputation enjoyed by professional engineers. Most of them would agree that we have not succeeded.

Traditional professions, such as engineering, law, and medicine, have

- a licensing authority, which identifies and certifies individuals who are competent to practice the profession and takes action against practitioners who either lack the required capabilities or do not practice properly
- demand-side legislation, which states that certain services and products can only be provided by licensed professionals

*Digital Object Identifier 10.1109/MC.2021.3057685*
*Date of current version: 7 May 2021*



0018-9162/21©2021IEEE

- an accreditation authority, which reviews educational programs and approves those that teach the capabilities required for licensing.

Licensed professionals usually display a certificate showing that they are licensed; they may also display their diploma(s) from accredited higher education institutions.

## PROFESSIONAL ENGINEERING: AN ABBREVIATED HISTORY

Licensing authorities were introduced when it became clear that people or organizations that need engineering services might not be able to distinguish individuals who were qualified to do the job from others offering to perform that service.

Demand-side legislation was introduced because legislators became aware that the public could be endangered when a service provider or product designer was not qualified to do the job.

Accreditation authorities were created because the licensing authorities, realizing that the education of an applicant for a license is a key determinant of the applicant's capabilities, found it more efficient to evaluate programs than to evaluate the education of each applicant. Accreditation also helps prospective students to pick a program that is likely to bring them closer to their career goals.

## BODY OF KNOWLEDGE OR BODY OF CAPABILITIES?

Many of the efforts to establish a software engineering profession have proposed a "body of knowledge" (BoK) for that field. A BoK is useful for characterizing a science because a science is an organized body of knowledge. Engineering is different. As Theodore von Kármán, a famous Hungarian-American mathematician and aerospace engineer, said, "Scientists discover the world that exists; engineers create the world that never was." Engineering requires knowledge, but it also requires the ability to apply that knowledge when designing. Engineering and other professions are better characterized by the capabilities required of their practitioners than by the required knowledge.

> For more than 50 years, people concerned about software development have tried to make it a profession like medicine or civil engineering.

Efforts to identify a BoK for software engineering seem doomed to fail for several reasons.

- The set of concepts and facts known to software developers is huge; there is little agreement on the importance and usefulness of even the most popular concepts. The size of the BoK has caused some developers to try to prioritize the knowledge and identify a required subset, but that is very difficult. What seems important to some seems useless to others.
- The software BoK is always rapidly growing; further, much of it becomes irrelevant just a few years after it has been added.
- Much of that knowledge is tied to specific tools. The characteristics of those tools are the result of many arbitrary design decisions, and the tools may evolve or become out of date; consequently, some of the knowledge about those tools will not be of lasting value.

To make sure that the graduates can have lengthy productive careers, the capabilities taught must be fundamental and of lasting value. There is a small set of basic principles and concepts that can be applied by software developers, but they are abstract and often hard to apply. The ability to use those principles must be taught; that is why the field is better defined by a body of capabilities rather than by a BoK. An attempt to identify the capabilities required of software engineers can be found in the article by Landwehr et al.[3] These capabilities include much more than the ability to write programs well.[2]

## SOFTWARE DEVELOPMENT IS NOT NOW AN ENGINEERING PROFESSION

Failure to agree on a suitable list of required capabilities has made it impossible for accreditation and licensing authorities to do their jobs. A lack of demand-side legislation makes any progress on licensing software developers almost inconsequential. Usually, the competence of a software developer is judged by an employer or customer with no help from a licensing authority.

## THE NEED FOR CHANGE IS URGENT

The world needs a revolution in software development. The quality of most software produced today is simply terrible. Our phone systems and computer networks are easily compromised. One of the world's richest men had his phone hacked although he is a computer expert. Data that are supposed to be confidential are frequently stolen. Bad software design

in the Boeing 737 MAX is blamed for two crashes with hundreds of deaths; a huge number of these new planes were out of service for about two years as a result. Financial and other government system projects frequently fail; some are put into service, with known flaws, long past their due date. Although we hear of software issues every week, many more are not reported to the public.

Many software developers are graduates of educational programs that did not give them the necessary capabilities. On-the-job training often teaches them the bad habits of older developers. Those habits are hard to break.

## ACADEMIC FREEDOM AND PROFESSIONAL EDUCATION

One of the most valuable characteristics of our educational system is the right of academics to express ideas without risk of official interference or professional disadvantage. However, many academics have interpreted this "academic freedom" to mean that they can teach whatever they want. Those who hold this opinion sometimes oppose the basic tenet of professional education, namely that an accredited curriculum must give its graduates specified capabilities. Basing professional accreditation on capabilities is a compromise. It allows academics to choose which facts, models, and methods they teach and how they teach those methods, provided that the graduates will have the required capabilities.

## LICENSING AND CIVIL LIBERTIES

Some believe that requiring those who want to practice a specific trade or profession to be licensed is a violation of their civil rights. They may even give an example, such as, "If I want to rewire a house for someone else, it is my right to do so." In fact, in many jurisdictions it is not legal to do so unless you are a licensed electrician. Our lawmakers, in their wisdom, have recognized that improper wiring can cause fires, expose the owners to risk of shock, and even damage power supplies. Requiring that people working as electricians be licensed does not violate our civil rights; it protects us. The same would be true if software developers required a license.

> Change will be strongly opposed because many people are doing very well with "business as usual" and will resent being told that they need to be reeducated.

## WHAT MUST BE DONE?

Changing software development into an engineering profession requires

- an agreed list of required capabilities such as that proposed in the article by Landwehr et al.[3]
- legally binding demand-side legislation
- a licensing authority with the legal power to enforce the demand-side legislation, license qualified developers, and discipline developers who do not practice properly
- an accreditation authority that reviews proposed professional software engineering programs to make sure that the graduates have the capabilities required for practicing the profession.

Those who want to establish capability-based licensing of software engineers must be prepared to face both inertia and bitter opposition.

The path to higher quality software requires improving the education and training of almost every software developer. Change will be strongly opposed because many people are doing very well with "business as usual" and will resent being told that they need to be reeducated. They will point to new requirements and techniques and proclaim, "Nobody does it that way."

Most of those who are teaching computer science today were educated in programs that were not designed to prepare students for licensing. Many will not see the need for the changes and will resent any move that might prevent them from teaching their favorite topics.

Some employers will not like the fact that licensed professional engineers are supposed to put public safety before employer profit. It is far easier to manage people who believe that their job is solely to please their employers.

However, our society is far too dependent on software being trustworthy to allow the present "Wild West" of software development to continue.

## REFERENCES


1. P. Naur and B. Randell, Eds., *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7–11 October, 1968, Scientific Affairs Division, NATO, Brussels*. New York: ACM, Jan. 1969.
2. D. L. Parnas, "Structured programming: A minor part of software engineering," in *Proc. Int. Workshop at the European Joint Conf. Theory Pract. Softw. (ETAPS'03)*, Apr. 6, 2003, pp. 19–25.
3. C. Landwehr et al., "Software systems engineering programmes: A capability approach," *J. Syst. Softw.*, vol. 125, pp. 354–364, Mar. 2017. doi: 10.1016/j.jss.2016.12.016.



**DAVID LORGE PARNAS** is a professor emeritus at McMaster University, Hamilton, Ontario, Canada, and the University of Limerick as well as president of Middle Road Software, Inc., Ottawa, Ontario, K1V 1V5, Canada. He is a Fellow of IEEE and ACM. Contact him at parnas@mcmaster.ca.