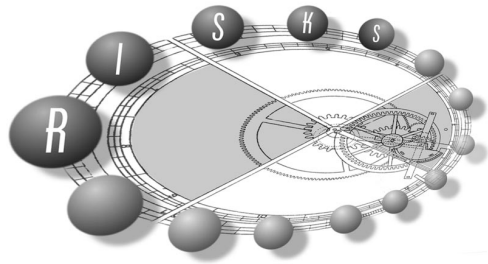

David Lorge Parnas

Software Engineering: An Unconsummated Marriage

When discussing the risks of using computers, we rarely mention the most basic problem: most programmers are not well educated for the work they do. Many have never learned the basic principles of software design and validation. Detailed knowledge of arcane system interfaces and languages is no substitute for knowing how to apply fundamental design principles.

The year-2000 problem illustrates my point. Since the late 1960s, we have known how to design programs so that it's easy to change the amount of storage used for dates. Nonetheless, thousands of programmers wrote millions of lines of code that violated well-accepted design principles. The simplest explanation: those who designed and approved that software were incompetent!

We once had similar problems with bridges and steam engines. Many who presented themselves as qualified to design, and direct the construction of, those products did not have the requisite knowledge and discipline. The response in many jurisdictions was legislation establishing engineering as a self-regulating profession. Under those laws, before anyone is allowed to practice engineering, they must be licensed by a specified professional engineering association. These associations identify a core body of knowledge for each engineering speciality. Accreditation committees visit universities frequently to make sure engineering programs teach the required material. The records of applicants for a license are examined to make sure they have passed the necessary courses. After acquiring supervised experience, applicants must pass additional examinations on the legal and ethical obligations of engineers.

When NATO organized two famous conferences on software engineering three decades ago, most engineers ignored them. Electrical engineers, interested in building computers, regarded programming as something to be done by others—either scientists who wanted the numerical results or mathematicians interested in numerical methods. Engineers viewed programming as a trivial task, akin to using a calculator. To this day, many refer to programming as a "skill," and deny that engineering principles must be applied when building software.

The organizers of the NATO conferences saw things differently. Knowing that the engineering profession has always been very protective of its legal right to control the use of the title "engineer," they hoped the conference title would provoke interest. They recognized that:

- Programming is neither science nor mathematics. Programmers are not adding to our body of knowledge; they build products.
- Using science and mathematics to build products for others is what engineers do.
- Software is a major source of problems for those who own and use it. The problems are exactly those to be expected when products are built by people who are educated for other professions and believe that building things is not their "real job."

Unfortunately, communication between engineers and those who study software hasn't been effective. The majority of engineers understand very little about the science of programming or the mathematics that one uses to analyze a program, and most computer scientists don't understand what it means to be an engineer.

Today, the problems that motivated the engineering legislation are rampant in the software field.

Over the years, engineering has split into a number of specialities, each centered on a distinct area of engineering science. Engineering societies must now recognize a new branch of engineering—software engineering—and identify its core body of knowledge. Just as chemical engineering is a marriage of chemistry with classical engineering areas such as thermodynamics, mechanics, and fluid dynamics, software engineering should wed a subset of computer science with the concepts and discipline taught to other engineers.

Software engineering is often treated as a branch of computer science. This is akin to regarding chemical engineering as a branch of chemistry. We need both chemists and chemical engineers but they are very different. Chemists are scientists; chemical engineers are engineers. Software engineering and computer science have the same relationship.

The marriage will be successful only if the engineering societies, and computer scientists come to understand that neither can create a software engineering profession without the other. Engineers must accept that they don't know enough computer science. Computer scientists must recognize that being an engineer is different from being a scientist, and that software engineers require an education very different from their own. C

DAVID LORGE PARNAS (P. Eng.) studies and teaches software design at McMaster University, Hamilton, Ontario, Canada.

PAUL WATSON