Software Engineering, What Must Be Taught

David Lorge Parnas[1]

Abstract

Software has become a critical technology in the modern world. Unfortunately, most of it can be characterized as unprofessional. Its developers have not been taught how to apply what is known about software development; they are often self-taught people who depend on intuition and use poor software as models. Consequently, software is famous for its flaws.

This talk is based on simple observations.

- Scientists add to, and organize, knowledge;
- Engineers apply science and mathematics to produce products that are fit for use by others.
- Engineers must be judged by what they can do (capabilities) rather than what they know.

This talk addresses the question, "What should a Software Engineer be able to do?"

The talk is addressed to educators, managers, licensing authorities, and, of course, developers.

[1] After lengthy discussions with Carl Landwehr, Jochen Ludewig Robert Meersman, Peretz Shoval, Yair Wand ,David Weiss and Elaine Weyuker whose contributions were substantial and substantive.

What Do Software Engineers Need To Be Taught How To Do?

- Communicate precisely between developers and stakeholders
- Divide work and communicate precisely among developers
- Design human-computer interfaces
- Design and maintain multi-version software
- Design software for reuse
- Revise old programs
- Software quality assurance
- Develop secure software
- Create and use models in system development
- Specify, predict, analyze and evaluate performance
- Be disciplined in development and maintenance
- Use metrics in system development
- Manage complex projects
- Deal with concurrency
- Understand and use non-determinacy
- Apply mathematics to increase quality and efficiency.

Properties Of The Capabilities On The List

They are all about "How to do it right" topics.

- "How to do it right" requires deep understanding.

They all involve some computer science.

They involve some (simple) mathematics

They involve understanding of science (knowledge organization)

They are not Science or Mathematics - they are capabilities.

They are fundamental — not technology.

Technological change will not invalidate this list.

We did not evolve to handle the complexity of software systems.

Learning to build them must be taught.

Reading Suggestions

1. Parnas, D.L., "Education for Computing Professionals", IEEE Computer, vol. 23, no. 1, January 1990, pp. 17-22.
2. Parnas, D. L., Chik-Parnas L., "Goals for Software Engineering Student Education", ACM SIGSOFT Software Engineering Notes, Volume 30 Number 4, July 2005, p. 6.
3. Landwehr, C., Ludewig, J., Meersman, R., Parnas, D.L., Shoval, P., Wand, Y., Weiss D., Weyuker E., "Software Systems Engineering programmes: a capability approach", in Journal of Systems and Software, Vol. 125, March 2017, pp. 354–364, Article: JSS9898 doi> 10.1016/j.jss.2016.12.016