



## **PLSE 2012 Panel: "Fixing Software Engineering"**

David Lorge Parnas

**This panel addresses the question, “What do we need to change to give Software Engineering the status of other Engineering Disciplines?”**



## **How Is Software Engineering Different From Other Disciplines?**

- **Physics and Chemistry not as relevant - but not irrelevant**
- **Software developers do not use mathematics in the same way but should**
- **Software developers confuse technology with science but shouldn't.**



## **Software Engineering Compared To Older Engineering Disciplines?**

**Engineers are expected to use technology, science and mathematics to create products that others can use and depend on.**

- **This should be equally true of those who develop software today.**

**Engineers are expected to leave records that allow others to review, use, maintain, and revise their products.**

- **This should be equally true of those who develop software today.**



## **Are There Different Types Of Software Engineering?**

**Are there different types of Of Software Engineers?**

**In all Engineering areas there are many application areas**

- **E.g., bridge building, road building, water supply building**
- **All have a common core, speciality comes later.**

**For Software Engineering we can see two broad areas:**

- **Engineering Applications implemented with digital technology.**
- **Information systems: - no physics involved**

**These have a lot in common (common core).**

**I advocate two “flavours”**

- **Professional Software Developer (not Engineers)**
- **Software Intensive Engineering (licensed Engineers)**



## **How Has Software Development Changed In The Past 40 Years? I**

**More people describe themselves as “software engineers”.**

- A nearly meaningless title.
- We do not know who knows what.

**The hardware is much more powerful**

- Things that were hard to do are now relatively easy (e.g. windows)
- Users depend on this.

**User interfaces are very different; feedback is nearly immediate.**

- Allows (encourages) hasty work and sloppiness.
- Our work depends much more on the work of others.
- Programmers often do not know the implications of their decisions.

**Concurrency has become an everyday problem - impossible to avoid**

- Still a source of bugs; much early work forgotten or ignored.

**Many almost standard interfaces that evolve quickly.**

- “Trial and error” has replaced disciplined use of documentation
- Constant need for updates.



## **What Are The Qualities We Would Like From Software Engineering In Order For It To Be A True Engineering Discipline?**

**Agreement on a “Core Body of Knowledge” (including old basics).**

**Licensing with recognition of specialities.**

**More knowledge of physics and control theory.**

**Habitual use of mathematics where it helps.**



## **How Has Software Development Changed In The Past 40 Years? - II**

**“Motorized-Mitten-like” development languages and environments.**

**Technology has replaced science**

- “Can’t be done with Java” replaces “Can’t be done”.
- Hiring and firing based on language.

**Updates are much easier**

- Needed more often. Early release “does no harm”.

**Interfaces “happen”. Details determined by the one who got there first.**

- Terribly bad design of interfaces.

### **Which differences are beneficial and which detrimental?**

**Hardware advances have made software a very powerful technology.**

**Newer languages have made many things easier.**

**Other changes have made the products less reliable.**



## **What Can Be Done To The Theory, Practice, And Education Of Software Engineering To Bring It Those Qualities?**

**Accreditation**

**Distinction between science and professional (Engineering) degrees.**

**Professional Style Education**

- Uniform cohort (few choices in lower years)
- Apply everything you learn.
- Cumulative project at end to use everything they learned (competitive).
- Clear distinction between technology and science
- Know how to learn a new language (more important than language choice).
- Know how to learn a new support system (more important than choice)
- Know how to work without a support system.
- Teach software equivalent of “blueprints” and other documentation.