Software Engineering

CS305, Autumn 2020

Software Engineering

Software + Engineering

What is Software?

- An abstraction that:
  - Defines a set of computations
  - Becomes concrete/useful only in the presence of hardware and context (e.g. human activity)

What is Engineering?

- *Traditionally:* "use of scientific principles to design and build machines, structures, and other items" - Wikipedia / Oxford dictionary

Why Software Engineering?

- Why is it so difficult to build software?
- Why is it so difficult to build good software?

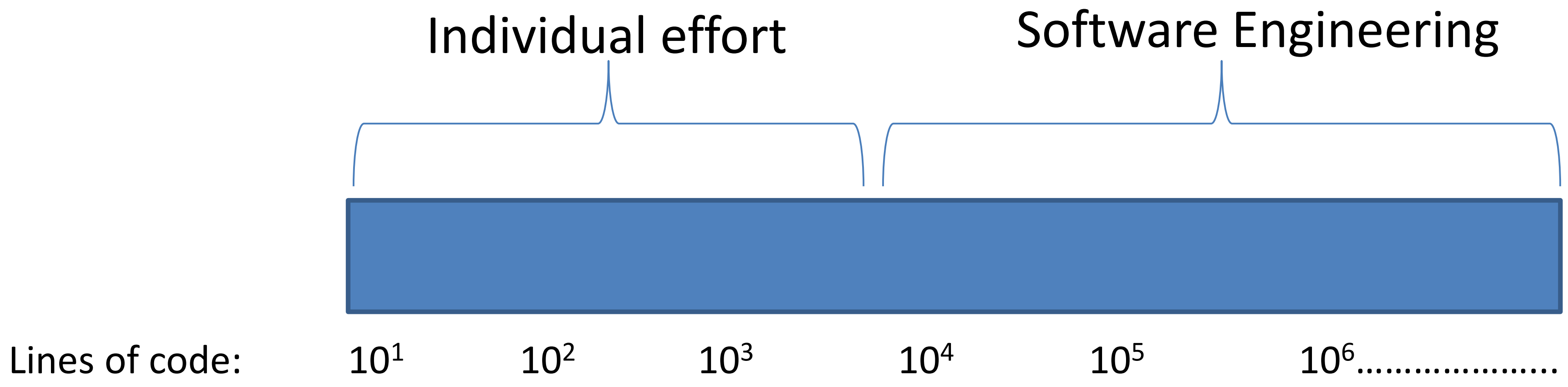
Software engineering is a fundamental discipline

Software Engineering

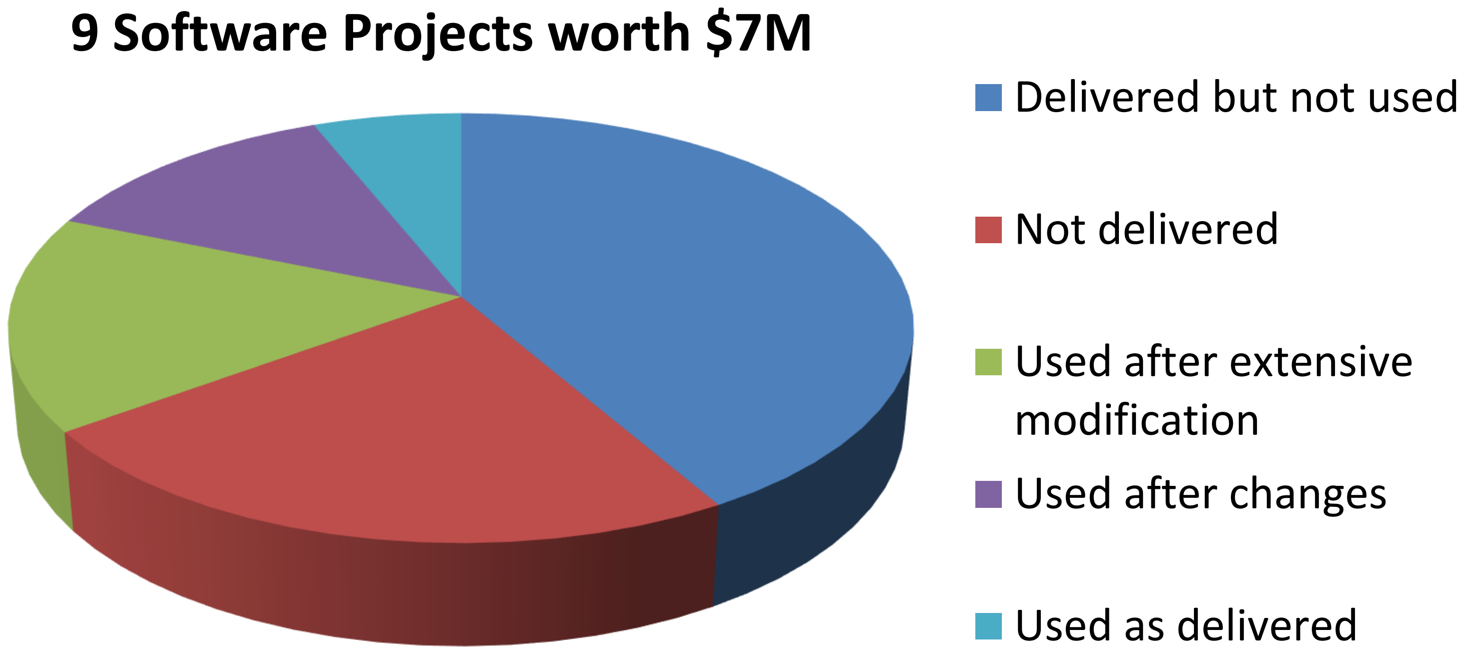
- Systematic study of:
  - Methodology
  - Techniques
  - Tools

to build high quality software that is *correct* and is built in a given *time and price budget*

Lines of Code in Software



Picture of a Crisis



- *$5M / $7M projects either not delivered or never used!*

Davis A. M. (1990) Software Requirements Analysis & Specification. Prentice-Hall, NJ.

Software Processes

- Transforming an idea to software is a complex task
- Processes help manage the complexity
  - Break the task into several steps/phases that are:
    - Systematic
    - Formal

Software Processes

- Transforming an idea to software is a complex task
- Processes help manage the complexity
  - Break the task into several steps/phases that are:
    - **Systematic**
    - **Formal**
  - E.g. 1) Waterfall model, 2) Evolutionary prototype 3) Unified Software Process, 4) Agile methodology

Exercise

- How many lines of code (LOC) does an average software developer produce per day?

LOC/day:

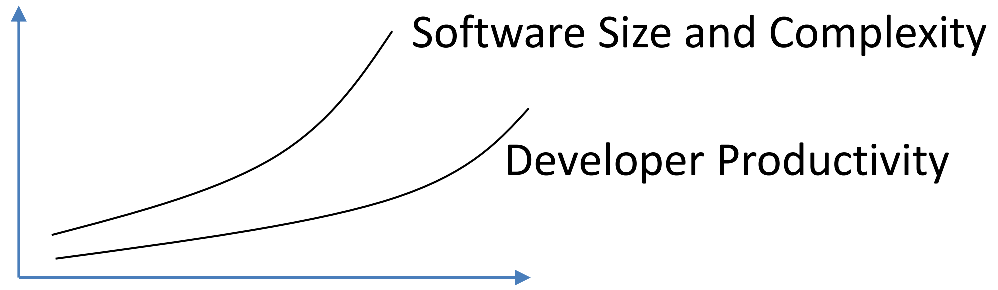
- ○ < 25
- ○ 25-50
- ○ 50-100
- ○ 100-1000
- ○ > 1000

Software Phases

- Processes are characterized by phases – steps in systematic software development
- Software Phases:
  1. Requirements / System Engineering
  2. Design
  3. Implementation
  4. Verification and Validation
  5. Maintenance

Tools for Software Engineering

- Software Complexity vs. Developer Productivity



- Productivity:
  - Development : punch cards vs. IDE (Atom, Eclipse, Microsoft Visual Studio)
  - Language: machine code vs. high-level language (e.g. C++, SQL)
  - Debugging: print statements vs. debuggers (e.g. GDB)
  - Others: Version control (e.g. Git), Code coverage and verification (e.g. Coverity, GCov)