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?

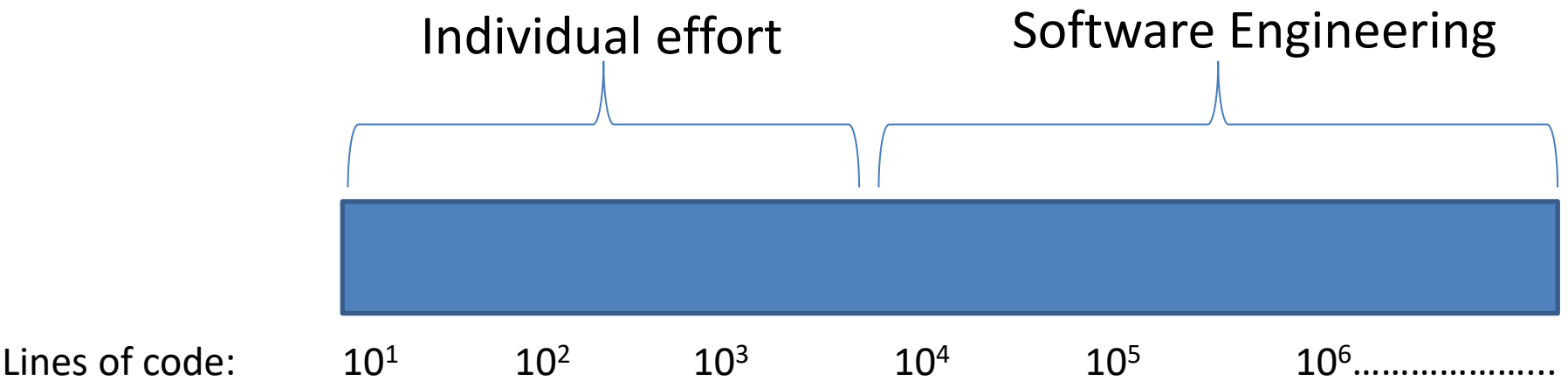
Software engineering is a fundamental discipline

Software Engineering

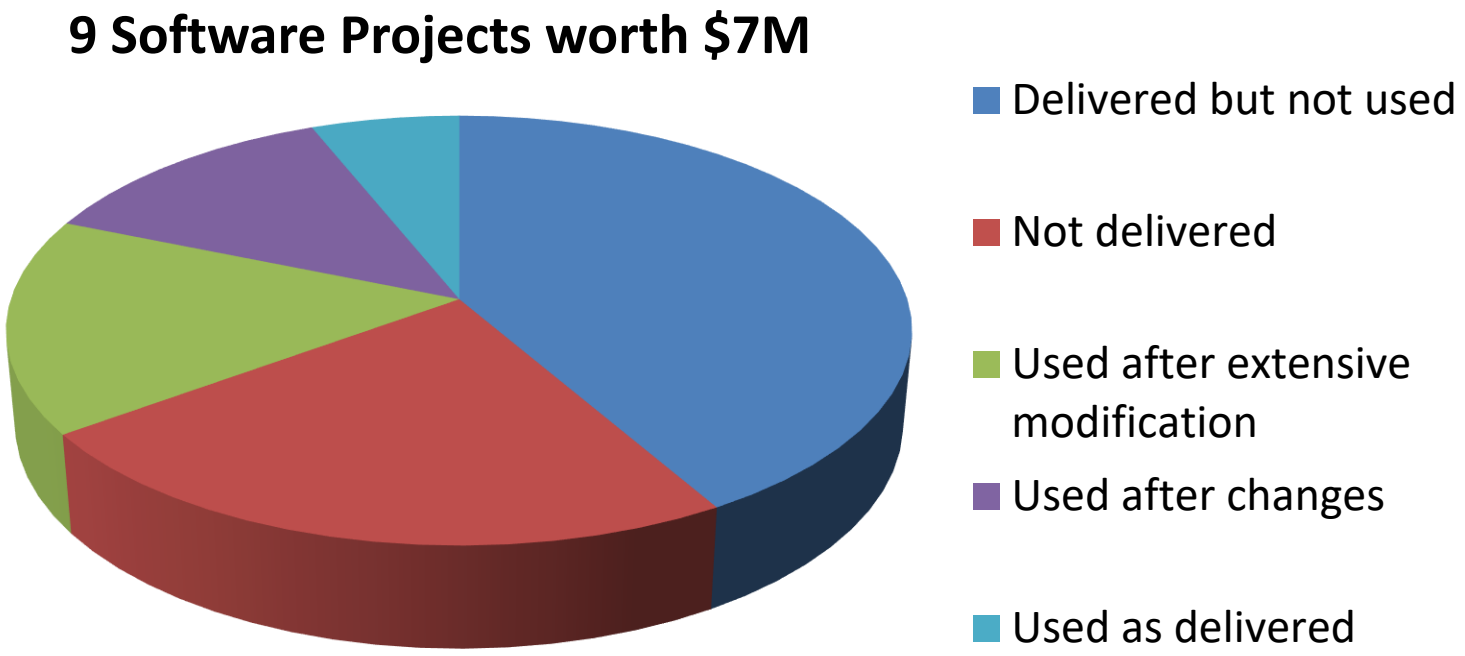
- Systematic study of:
  - Methodology
  - Techniques
  - Tools

  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
  - https://forms.gle/uyCqG6rMrnGavmhz8

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

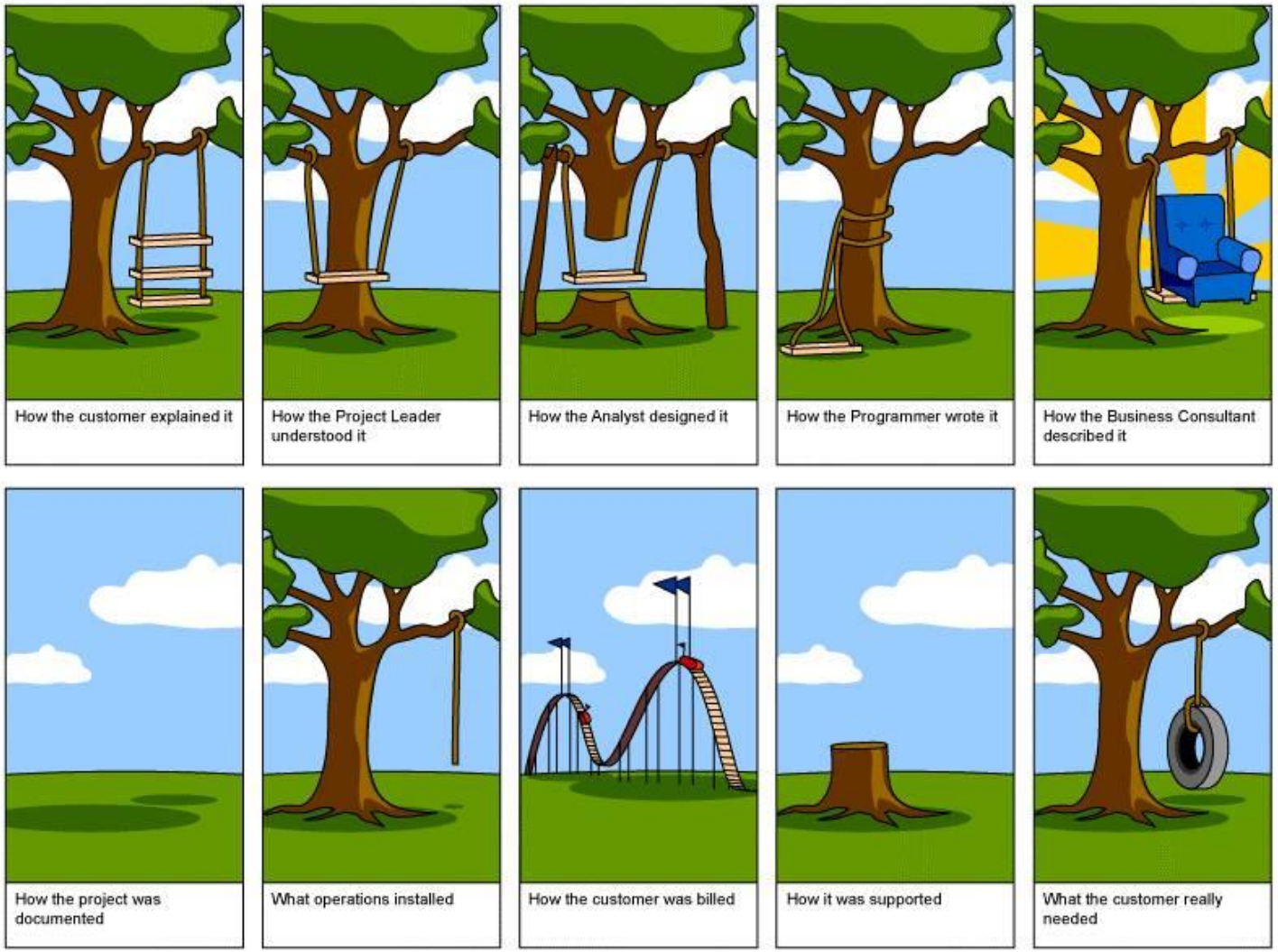
Last Class…

- What is Software Engineering ?
- What lead to Software Engineering as a discipline?
  - Software boom, Complexity, Failures
- Why is it so difficult to build good software?
  - Correctness, Price and Time constraints
- What are different phases or activities of software development?
  - Ordering of activities
  - Duration of activities

} Processes

Turning Ideas into Software

is complicated!



Source: Alex Orso (CS 3300)

Today's Class…

- Managing Software Complexity Through Software Development Life Cycle (SDLC) Models / *Processes*
  - Waterfall Model
  - Spiral Model
  - Evolutionary Prototyping
  - Rational Unified Process
  - Agile Methodology
- Software Complexity and Programmer Productivity

SDLC Activities / Steps – Requirements Engineering

- Establish stakeholders' needs that are to be satisfied by the software
- Why Important?
  - Cost of correcting errors
    - Grows exponentially as we move to maintenance phase
- How to get it right?
  - Elicit, Analyze, Specify, Validate, Manage - *Iterate*

SDLC Activities / Steps – Design

- Translate Requirements to internal structure
  - Architecture, Interface, Component, Data structure, algorithm

SDLC Activities / Steps – Coding

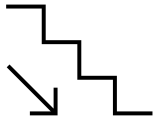
- Translate design into software
- How to get it right? Keep a tab on..
  - Complexity
  - Diversity
  - Validation
  - Standards

SDLC Activities / Steps – Verification and Validation

- Have we built the right system? - validation
- Have we built the system right? – verification
- Done at:
  - Unit, Integration, System levels

SDLC Activities / Steps – Maintenance

- Deals with handling issues / requests seen after the software is delivered
  - Corrective – Bug fixes
  - Perfective – enhancements
  - Adaptive – environment changes

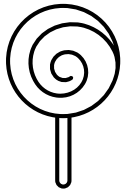


Waterfall Model

- Go from one phase to another like a cascading waterfall

  Software Concept ⟷ Requirements Analysis ⟷ Architectural Design ⟷ Detailed Design ⟷ Coding and Debugging ⟷ System Testing

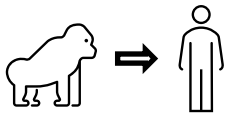
  - Very old
  - Pros: Finds errors easily
  - Cons: Not flexible



Spiral Model

- Incremental Risk-Oriented Model
  - Determine Objectives
  - Identify and Resolve Risks
  - Develop and Test Software
  - Plan Next Iteration

  Iterate…
  - Cons: Complex, Dependent on Risk Analysis, Requires Specific Expertise
  - Pros: Risk Reduction, Easy to Enhance, Software Produced Early



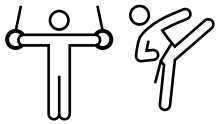
Evolutionary Prototyping

- Evolve an initial prototype based on customer feedback
  - Start with an initial Prototype
  - Design and Implement the Prototype
  - Refine Until Acceptable
  - Complete and Release

  - Pros: Immediate Feedback, Helps Requirements Understanding
  - Cons: Difficult to Plan, Can Deteriorate to Code-and-fix

Rational Unified Process

- Use Unified Modeling Language (UML) to formally capture the order and duration of different activities
  - Inception, Elaboration, Construction, Transition



Agile

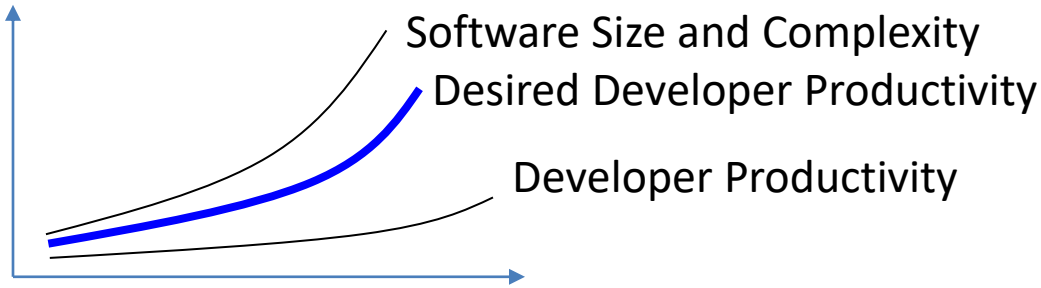
- Be more flexible (trading off discipline) in accommodating changes in requirements
  - Fail, Pass, Refactor

Choosing A Model

- Requirements Understanding
- Expected Lifetime
- Risk
- Schedule Constraints
- Interaction with Customers
- Expertise

Tools for Software Engineering

- Software Complexity vs. Developer Productivity



- Productivity:
  - Development : punch cards vs. IDE (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)