CS471 Lecture 01

Software Engineering Introduction and Motivation
Sommerville Ch2

What is Engineering?

What is Engineering?

- "the application of mathematics, science, economics, empirical evidence, etc. to invent, innovate, design, build, maintain, research, and improve structures, machines, tools, systems, components, materials, processes, solutions, and organizations. "

What is Software Engineering?

Software Engineering Definitions

- "...an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance"

  -Sommerville

- "...the application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software..."

  -IEEE

Software Engineering

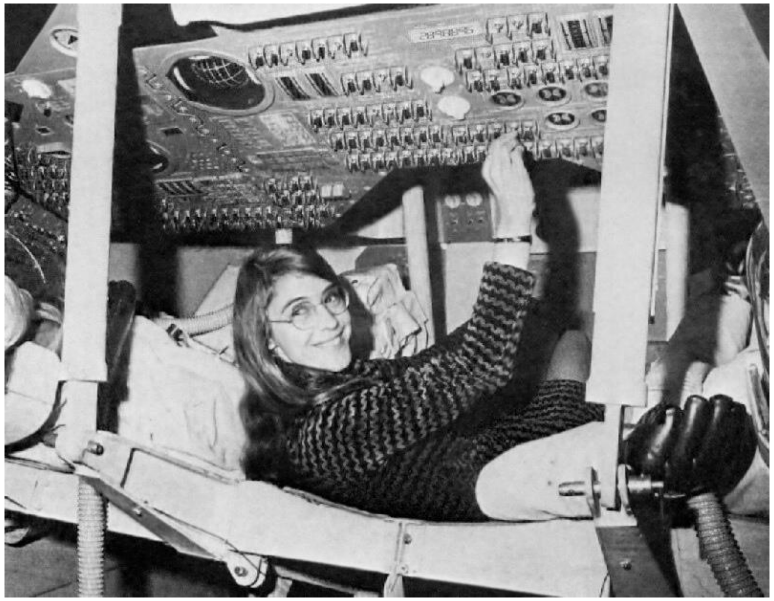
- First software (early 50's)
  - cost of hardware dominates
  - programs seem to be less important

Software Engineering

- First software (early 50's)
  - cost of hardware dominates
  - programs seem to be less important
- Software crisis (late 60's)
  - hardware becomes cheaper
  - custom software becomes complex and expensive
  - software production lags behind the need
  - software engineering discipline is born (early 70's)

Nascent Software Engineering Applications: Aerospace

- Margaret Hamilton, Lead Flight Software Designer, Apollo Program
- Prevented an abort of the Apollo 11 lunar landing



Nascent Software Engineering Applications: Aerospace

- See source code at:

https://github.com/chrislgarry/Apollo-11

- 80KLOC (i.e., 80,000 Lines of Code) written in Assembly

Original Software Engineering Objectives

- Improve the following competing resources
  - Quality
  - Schedule
  - Cost

- Largely focused on the development of large aerospace and enterprise applications

Questions about software

- Why does it take so long to get software completed?
- Why are costs so high?
- Why can’t all errors be found before the software is put into production?
- Why is it difficult to measure the progress at which software is being developed?

High-level Explanations to Questions about software

- Software is developed (or engineered), not "manufactured" (in the classical sense)

- Software does not "wear out" (as do traditional concrete products), but it "deteriorates" during requirements, design, development, maintenance

- Most software is custom built rather than assembled from existing components

Software Engineering vs. [other] Engineering

- One of the essential technologies of today
  - essential for economy
  - essential for security

- Technology of the same importance as
  - mechanical engineering
  - electrical engineering, etc.

- How does software and engineering differs from other engineering fields?

Software Engineering vs. [other] Engineering

- Other branches of engineering use standardized tools and metrics to produce systems with predictable outcomes

- Mechanical and electrical engineers have big catalogs of standard parts they recycle into their creations vs. “reinventing the wheel”

Software Engineering vs. [other] Engineering

- Other branches of engineering use standardized tools and metrics to produce systems with predictable outcomes

- Mechanical and electrical engineers have big catalogs of standard parts they recycle into their creations vs. "reinventing the wheel"
  - Trust / compatibility of existing software components?

Software Engineering vs. [other] Engineering

"Local" implications

"Global" implications

Properties of Software*

*Brooks, Fred P. (1986). "No Silver Bullet — Essence and Accident in Software Engineering". Proceedings of the IFIP Tenth World Computing Conference: 1069–1076

Properties of Software – Accidental

- Accidental properties change from time to time
- Examples:
  - Programming language
  - Hardware speed, memory size
  - Architecture of the program
    - functional
    - object oriented
- Solutions:
  - High-level programming languages
  - Time-sharing
  - Unified programming environments

Properties of Software – Essential

- Intrinsic to software – determine its nature
- These do not change!

- Complexity
- Conformity/Interoperability
- Changeability
- Invisibility
  - not tangible
  - cannot use senses