CS 370
REVIEW: Early History of Software Engineering

DR. MICHAEL J. REALE
FALL 2015
Pre-SE

- Early on, no “software engineering”
  - Batch processing $\Rightarrow$ slow development
  - Constantly changing hardware
  - Hardware-specific assembly language

- Increase in software development (60’s)
  - Higher-level languages $\Rightarrow$ portable
  - Interactive, time-sharing machines
  - More powerful computers
The “Software Crisis”

- “Software Crisis” → identified in the mid/late 60’s
  - Projects over time and budget (or never completed at all)
    - Poor estimation of how long things would take
    - Poor monitoring of progress
  - Projects poorly managed (or not at all)
    - Assumption that people and months are interchangeable
    - Adding manpower when behind schedule
  - Software did not meet requirements and/or was of poor quality
    - ...sometimes with fatal results... (e.g., Therac-25)
Software Engineering is Born

- 1968 – first NATO Software Engineering Conference (Germany)
  - Term “software engineering” coined
  - Trying to address “software crisis”
  - Recognized need to some formalized process of software development
    - Also called “software process”

Fred Brooks

- Dr. Frederick P. Brooks, Jr.
  - Project manager of IBM System/360 (hardware)
    - Coined term “computer architecture”
  - Manager of OS/360 software project
    - Experience resulted in book “Mythical Man-Month”
  - Won Turing Award in 1999

http://s3v.computerhistory.org/102695713-03-02.jpg
http://caem.acm.org/system/assets/0000/3115/060910_UNC_Fred-Brooks.large.jpg?1341312413&1276112307
The Problems with Software Development

- Brooks → why software projects fail:
  - Poor estimation
    - Unwarranted optimism
    - Need previous project data
    - Not enough time for testing/debugging
  - Assuming people and months are interchangeable
    - More people → more lines of communication
  - Adding manpower to a late project
    - **Brook’s Law:** *Adding manpower to a late software project makes it later.*
      - More lines of communication
      - More training time
      - Possible poor partitioning of remaining work
  - “Gutless estimating”
    - If project is late:
      - Rush it → bad idea
      - Tell customer it will be late
      - Offer alternative with fewer features
  - Poorly monitored progress