CS1530, LECTURE 19: TRADE-OFFS IN SOFTWARE ENGINEERING

BILL LABOON

IRON TRIANGLE OF PROJECT MANAGEMENT

- Scope / Quality
- Time
- Resources

OPTIMIZATION OF THE IRON TRIANGLE

- You can optimize according to time or resources
  - More people up front, trailing off
  - Stretching schedule to meet scope
  - Hiring more employees
- But remember Brooks' Law!
  - "Adding more people to a late software project makes it later"
  - There is not always a linear relationship between legs of the triangle

OPTIMIZATION OF SCOPE/QUALITY

- Oftentimes you have hard deadlines, or limited funds (e.g. for a class)
- Your only option is to optimize scope
- Two different aspects
  - Quality Attributes (e.g., security, usability, performance)
  - Functional attributes (features)
- Can you compromise on internal quality without compromising external quality?

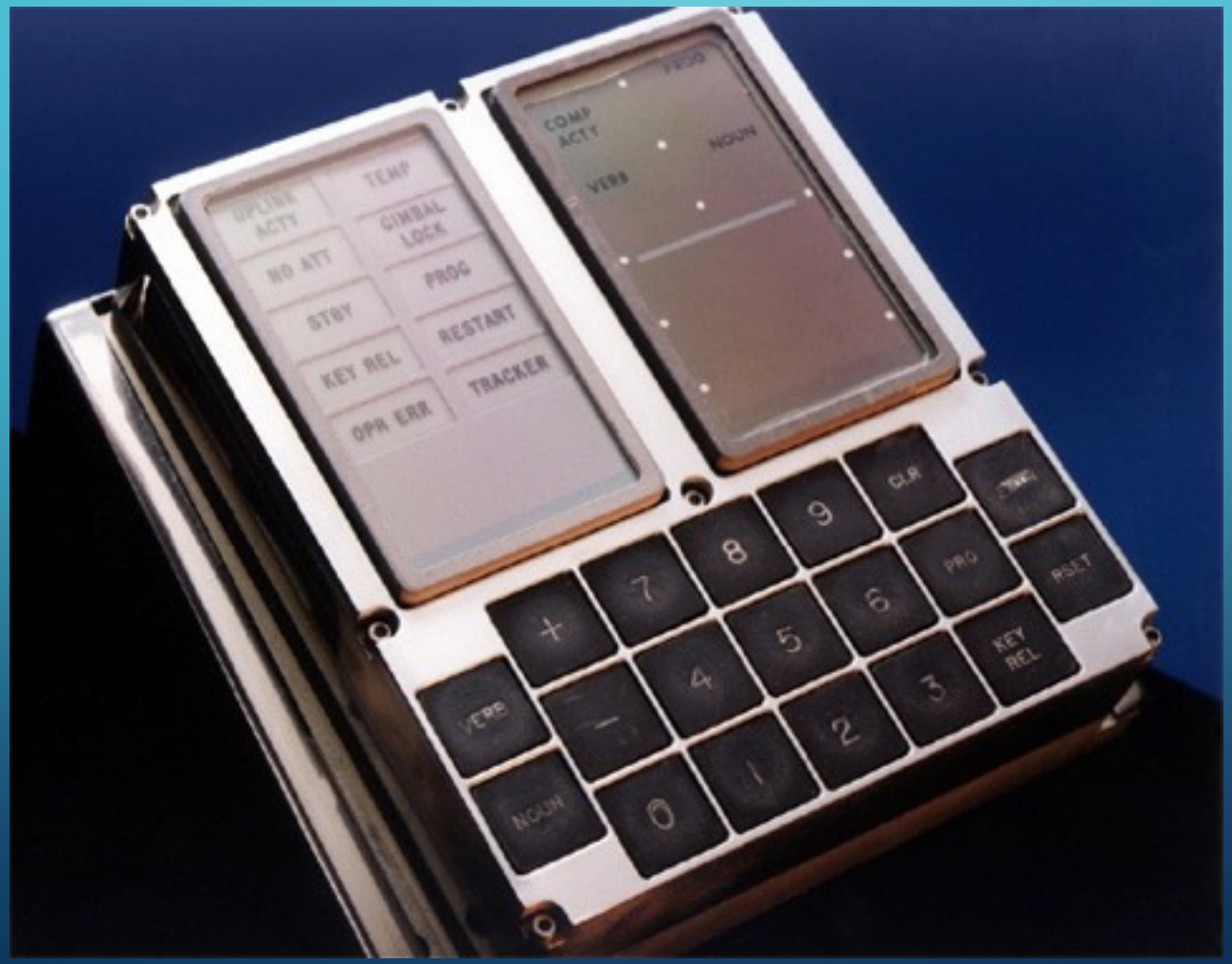
NECESSARY VS NICE-TO-HAVE

- If a feature will get you 10 additional users, is it worth delaying delivery or cutting other features? What about 100? A million? A billion?
- Is internal quality better than external quality?
  - What if we need to upgrade Super Mario Bros?!?!
  - Does portability or maintainability help us here?
- On the other hand.. would better internal quality have helped OpenSSL?
- Internal quality often leads to better external quality, but NOT ALWAYS
- Sometimes better to err to one side, sometimes the other.

MAKING A LIST, CHECKING IT TWICE, GONNA FIND OUT WHO'S NAUGHTY XOR NICE

- List and quantify benefits / drawbacks
- Come up with broad estimates of what's necessary
- Work with PMs and other stakeholders to determine relative importance
- In performance testing, there is the concept of threshold and target
  - What is your threshold of features? What is your target?

CASE STUDY: APOLLO GUIDANCE COMPUTER



CASE STUDY: APOLLO GUIDANCE COMPUTER

- Virtual AGC info - http://www.ibiblio.org/apollo/index.html
- AGC UI interface - http://svtsim.com/moonjs/agc.html
- https://www.linux.com/news/software/developer/29068-apollo-11-story
- AGC code: https://github.com/chrislgarry/Apollo-11/
- *Digital Apollo* by David Mindell

TRADE-OFF: INTEGRATED CIRCUITS OR TRIED-AND-TRUE TRANSISTOR TECHNOLOGY?

- Transistors had been around since the early ‘50s, only one company (Fairchild Semiconductors) was making the appropriate ICs
  - What if Fairchild stopped making them?
- Easier to use ICs – made for a more modular system
- Twice as fast as the transistor model
- Used up twice as much wattage

RESULT: ICS

- Mitigations:
  - NASA started using ICs for other equipment
  - Standardized on one particular type of IC – could use the best of the bunch, ensured a market for that kind
    - At one point the Apollo program was using 60% of all ICs produced in the US!
  - A prototype was produced first to see that it was possible
  - Fairchild DID in fact stop producing the chips! But NASA had made a market and Philco continued to produce them

TRADE-OFF: BARE-METAL OR INTERPRETED CODE?

- Bare metal would provide ultimate control over system
- Additionally, speed and memory were at a premium
  - CPU ran at 1.024 MHz
  - 32 kilowords (64 kilobytes) of read-only memory
  - 4 kilowords (8 kilobytes) of read/write memory
- Virtual machine would allow more safety and understandability of code

RESULT: INTERPRETED CODE!

- Sophisticated RTOS (real-time operating system)
  - Allowed safety, resetting of stalled tasks, etc.
- Could do complex commands (e.g. matrix arithmetic, trigonometric functions) mixed in with low-level assembly
- Never underestimate how hard it is to write software! Readable code is VERY valuable (see the slides on Software Craftsmanship)

CASE STUDY: ADDING METRICS

- Teachers wanted statistics on students using our software
- Examples: What percentage of students did work in the last week? How many questions did they answer? What grade level are most students working at?

TRADE-OFF: EXCEL SPREADSHEET/CSV OR PRETTY GRAPHICS?

- Spreadsheet would allow more power and analysis on their part
  - But it might be hard to understand
- Graphics would be very pretty and easy to understand
  - Hard to do deeper analysis

RESULT: GRAPHICS

- After discussing with product managers, few teachers needed or even WANTED to do deep analysis
- Their goal was to see an at-a-glance high level view of the class, not calculate the skewness and scedasticity of certain variables

TRADE-OFF: PERFORMANCE

- Running these reports on individual classes was time-consuming
  - A teacher would ask for them, and it could be several minutes before the page popped up
- Choice: spend engineering time to optimize it, or run it overnight once a week and pre-populate?
  - Former: Teachers would get most up-to-date information, but we would spend valuable engineering time to do it. It would also impact performance for students or cause to buy more CPU power.
  - Latter: We would always be out-of-date by up to a week. We would be calculating data even for teachers who never used the tool, since we'd have to prepopulate everything.

RESULT: PRE-POPULATE EVERYTHING

- Engineering time was paramount – we weren't sure how long it would even take
- Performance was already not great – did not want to impact it further
- Buying CPU resources at night and allowing it to be done whenever (low priority) was much cheaper than doing it on-demand (high-priority)
- Most importantly, it was not a big issue to teachers – weekly status reports were granular enough for them

SOFTWARE ENGINEERING IS THE STUDY OF TRADE-OFFS