Agile Software Development I
Rapid iterations for fun and profit
Overview

Why should we care about process?
History: the waterfall model
What’s particular about developing software?
The agile model

Why should we care about process?
We want to solve big problems!
- launch a rocket
- reduce energy usage in my office
- talk to relatives across the globe

We want to make sure we get it right.
We can’t do it alone (too large, too complex).

History: software engineering process was adapted from other disciplines.

Characteristics of physical engineering projects
Requirements and technologies are (relatively) stable.
Very costly to change course, even more so to fail.
- Very expensive to change the foundations of a building after the fact
- NASA shuttle software controls $4B piece of machinery

Development time is not the primary concern.
- I, for one, would rather be a little late than CRASH THE $4B ROCKETSHP

Result: the waterfall model
Development process is organized into a series of sequential phases that flow towards delivery of a complete product.
The waterfall model

*Measure twice, cut once.*

What are the benefits of the waterfall model?

- Mistakes minimized due to investment in up-front design.
  - e.g. over 11 versions of NASA shuttle software (each 420,000 LOC) just 17 errors. Commercial software of the same scale would have ~5000.
  - Very important for high-stakes software (medical, defense, etc.)
- Emphasis on requirements and specification leads to well-documented projects.
  - Easy to analyze and critique
  - Supports turnover and long timelines well

What sort of beast is software?

- (Full) Requirements are uncertain
  - Some pieces are clear, but full system is hard to specify at the start
  - Feedback is essential to getting it right
- Underlying technologies change rapidly
- Costs of change and failure are (often/relatively) low

In short: **change is expected and inevitable** (and maybe not so bad)

Here's a thought: in the face of change, time is the real enemy

- “More software projects have gone awry for lack of calendar time than for all other causes combined.”
- Taking a long time to do something in a rapidly changing world is inherently risky.

How well does the waterfall model measure up to this context?

- Lots of time and effort spent on planning and documentation that may never be relevant
- Long time to first delivery of functional software
- Little room for feedback once we've left the analysis phase
No more planning! (or design, or testing, or ...) Am I right?

Noope.

Still far cheaper to fix a problem in the design stage than during implementation.

Good software design and decision-making pays very big dividends (the "10x" developer)

So then, what’s the right compromise?

The agile model

Deliver (working) software early and continuously

Manage scope very carefully in order to achieve this

Recap

Process and context are inextricably linked.

In a lot of software contexts, agility (through rapid iteration) is a great process model