Engineering Design w/Embedded Systems

Lecture 24: Comparing Lifecycle Models

Patrick Lam
University of Waterloo

March 12, 2013

Design and Planning

Traditionally:

- solicit requirements;
- make a design;
- analyze the design (with calculus);
- stamp and sign the plan.

Someone else builds the plan.

People tried this for software as well:

- solicit requirements;
- make UML diagrams;

and hire code monkeys to implement the design.

This works poorly.

The Role of Prototyping

> *The management question, therefore, is not* whether *to build a pilot system and throw it away. You* will *do that. [. . .] Hence* plan to throw one away; you will, anyhow*.*
>
> Frederick Brooks, *The Mythical Man-Month*, 1975.

The biggest reason the waterfall is a straw-man:

- You can never build a system without a prototype, because:
- You never really know what you need until you see it.
- Also, needs change over time.

Solution: **iteration**.
Different models have different iteration strategies.

Agile Lifecycle Models

Key point: handling change on-demand.

Incorporate change into fast iteration process.

This requires suitable developers.

- Agile models are always collaborative.
  Hence, allegedly better at dealing with:
  1. changes to team over time; and
  2. differences between developer experience levels.

Choosing a Lifecycle Model

As we've said, we need to control the pace of iteration:

- Large teams, diverse stakeholder groups: slower iterations;
- Small teams, uncertain requirements, complex technologies: faster iterations.

Questions to Think About

- Do you understand customer requirements?
- Will you need to make major architectural changes?
- How reliable does the system need to be?
- How much future expansion and growth do you foresee?
- How risky is the project?

More Questions to Think About

- Is the schedule heavily-constrained?
- What is going to change during development?
- How much do customers need visible progress?
- How much does management need visible progress?
- What experience does the design team bring?

Software Process Improvement

Engineer the software development process:

- first, figure out what you mean to be doing—document your software process.
- next, figure out what you're actually doing—ensure that you're following existing process.
- finally, identify areas of potential improvement, and implement them.

Continuous process improvement: constant review.

Beyond the scope of an individual project, but rather across projects.

Allegedly: yield dramatic increases in development capability.