Project Discussion and Questions / Software Development Processes

CSCI-3081: Program Design and Development

What we’re seeing in your individual UML designs: This is Hard

- You have to look at our support code to start.
- You have to figure out (enough of) what’s happening there to plan a solution.
- You have a plan a solution.
- The solution is complex enough that you can’t (and shouldn’t) start working on it right away by typing in an editor — you need to plan and design.
- You’re doing that (initial) design using a new language — UML.
- This is probably the first time you’ve been asked to do most of these things.

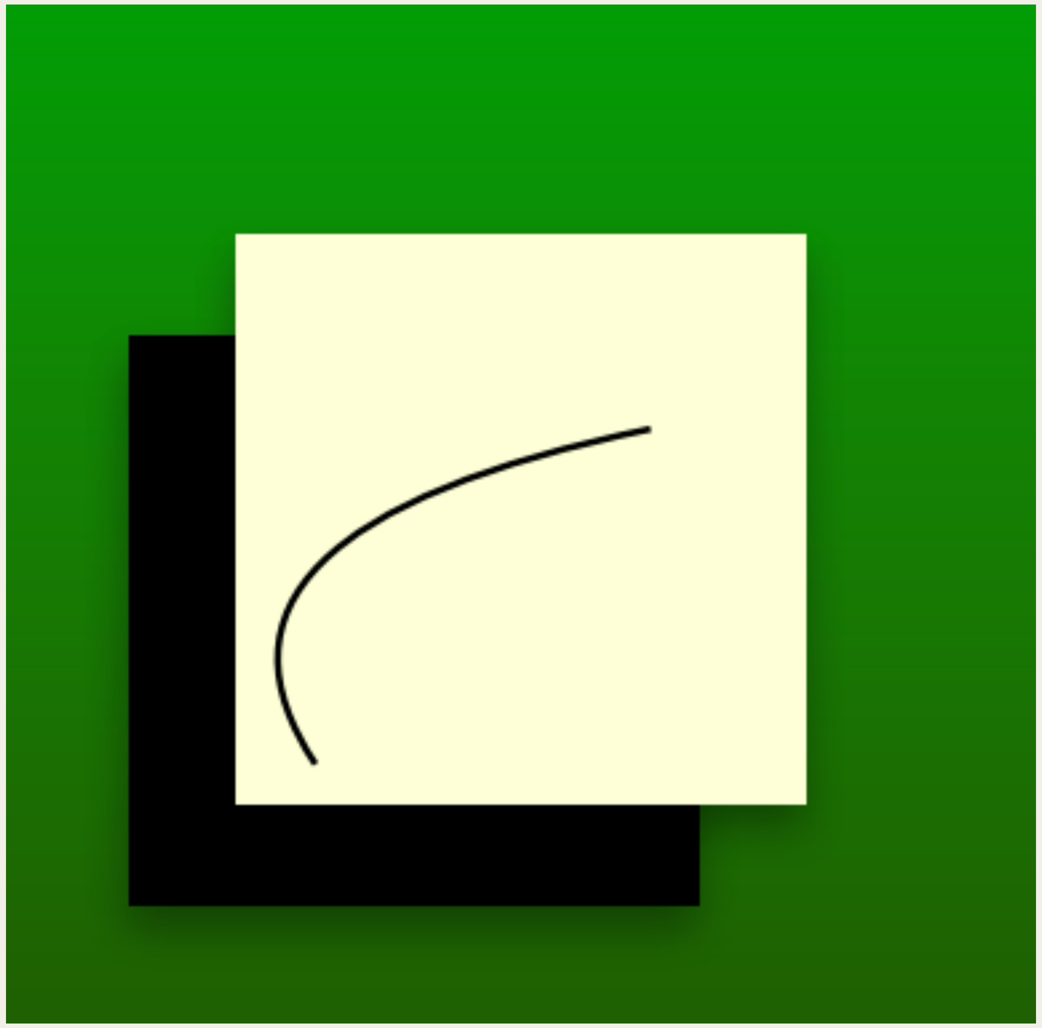
Ok it’s hard, how do I get through it?

- Stick with it.
- Enjoy this great learning opportunity — (I think) we’ve already convinced you that this experience really mirrors what software development is like outside of school. This is really valuable.
- Work with each other and help each other in your groups.
- Ask questions now and in office hours.

Brushwork F.A.Q.

What is Alpha?

- Transparency, stored as the fourth vector component of a color (R, G, B, A).
- We want to keep alpha = 1.
- If alpha is < 1, black will show through our canvas.
- You should never need to set alpha to anything other than 1 for this iteration.



How do we blend colors?

- We need to take into account both the tool color and the current canvas color.
- Let’s look at just the red component:

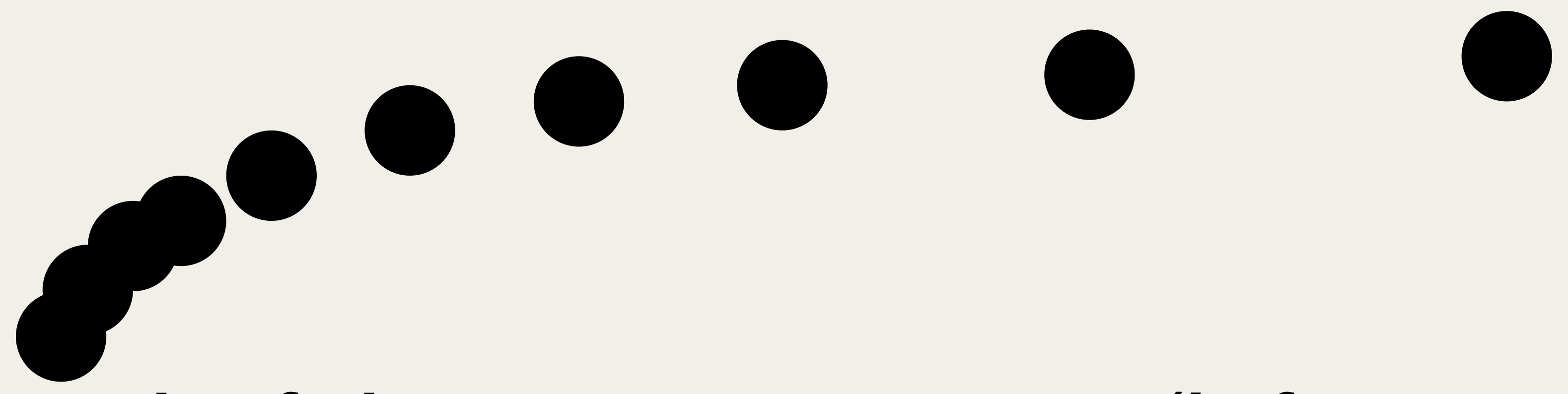
```
canvasColor.setRed(toolColor.getRed()*intensity +
                   canvasColor.getRed()*(1.0-intensity));
```

- You can do the same thing for Green and Blue. Or, use the multiply operator that is built in to ColorData as a shortcut:

```
canvasColor = toolColor*intensity + canvasColor*(1.0-intensity);
```

- Notice the blending equation keeps alpha = 1, and R,G,B end up being a weighted average of toolColor and canvasColor.

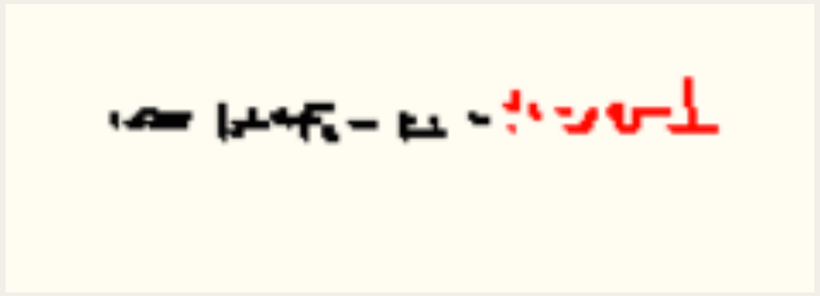
How do we make the pen look more continuous?

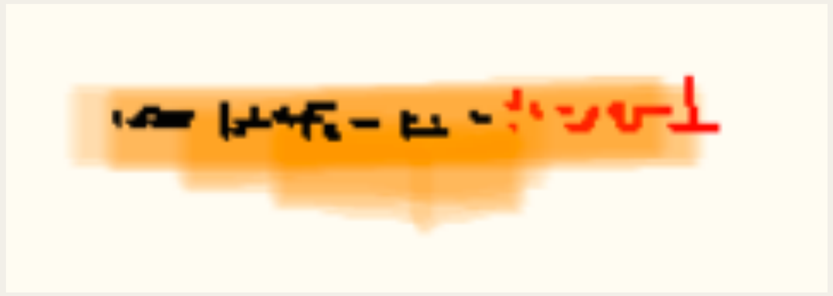


speed of drawing increases (left to right)

- [Absolutely required] Use a mask to avoid unnecessary computation — this helps the display to update faster.
- [Required, but less stringent] In general, use good software design with an eye toward writing efficient code.
- [Not required, but you're welcome to do if you want] Fill in the gaps between mouse move events.

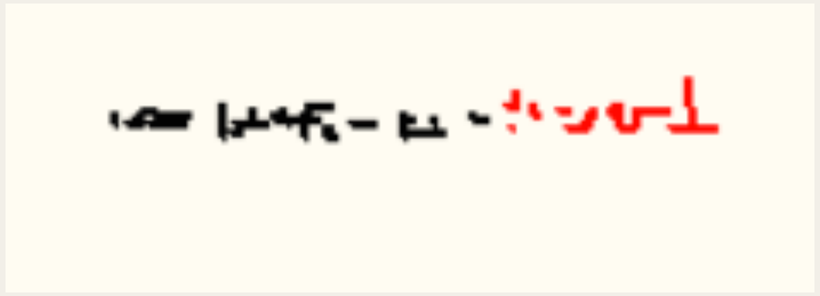
How to make a highlighter that really looks like a highlighter (not strictly required, but FYI):





- If you have made it far enough into the project to work on the highlighter, it is perfectly fine if you follow the strategy described in the assignment handout (just a simple 40% highlighter color, 60% canvas color blend).
- But, if you want a highlighter that acts a bit more like a highlighter, here is a neat way to do the blending to make that work…

How to make a highlighter that really looks like a highlighter (not required, but FYI):



- ColorData has a function called getLuminance() that returns a measure of how bright a RGB color is (note, green light contributes the most to the intensity of color as perceived by humans):

```
   L = (0.2126)R + (0.7152)G + (0.0722)B;
```

- Use this to modulate the blending so that the dark pixels show through the highlighter a bit more:

```
intensity = maskAmount * canvasColor.getLuminance();

canvasColor = toolColor*intensity + canvasColor*(1.0-intensity);
```

Additional Questions (now is your time)

(Formal) Software Development Processes

Motivation for Software Development Processes

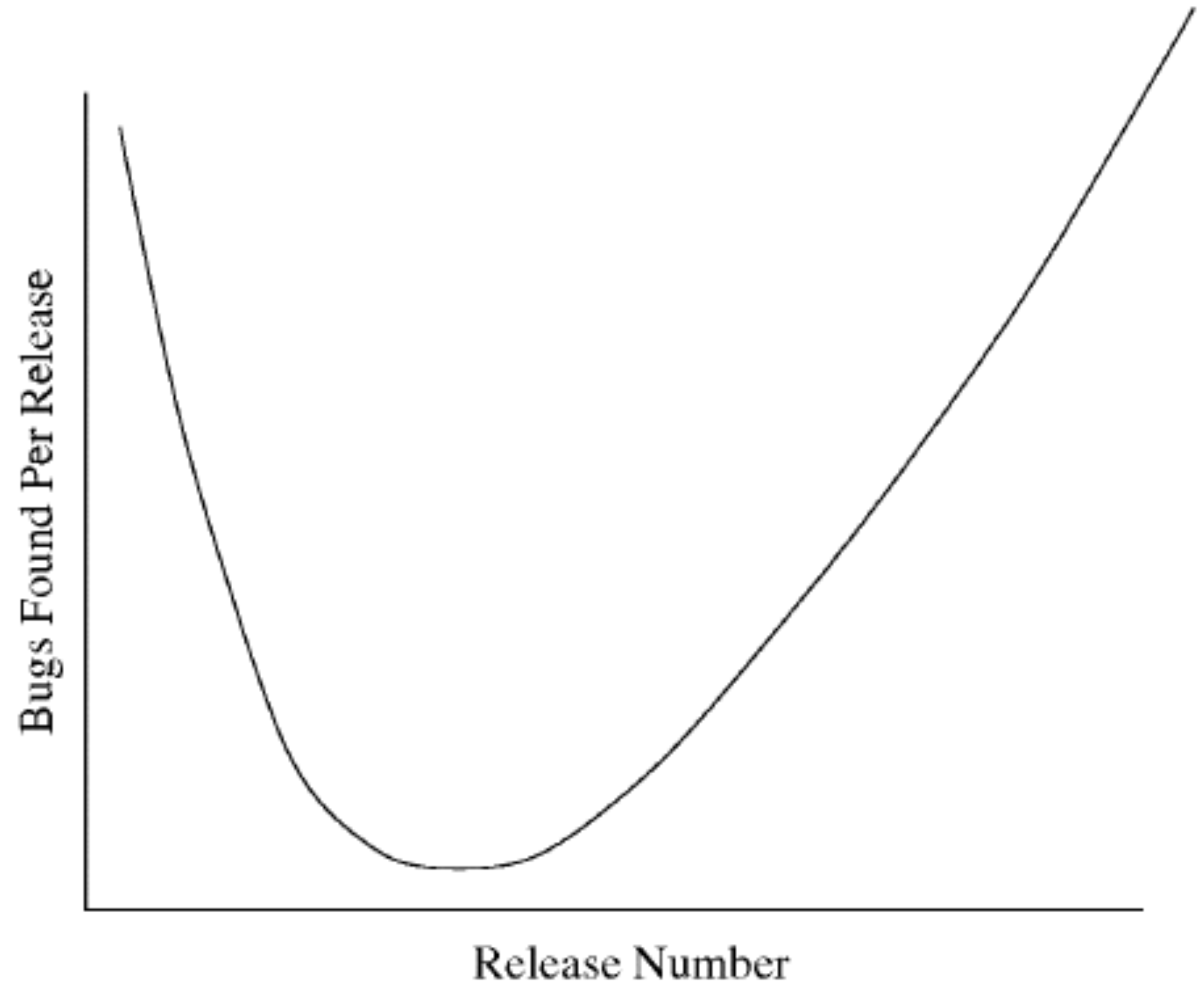


Figure 1: Belady-Lehman Graph

Understanding Software Development

- Metaphors -- McConnell suggests several.
- Here's one that caught my eye (McConnell, p. 16)
    - Build a 4 foot tower out of 10 beer cans -- not too difficult.
    - Build a tower 100 times that size -- doesn't just require 100 times as many beer cans, requires a different kind of planning and construction entirely.
    - See: http://www.youtube.com/watch?v=jp8jIPaX9BM

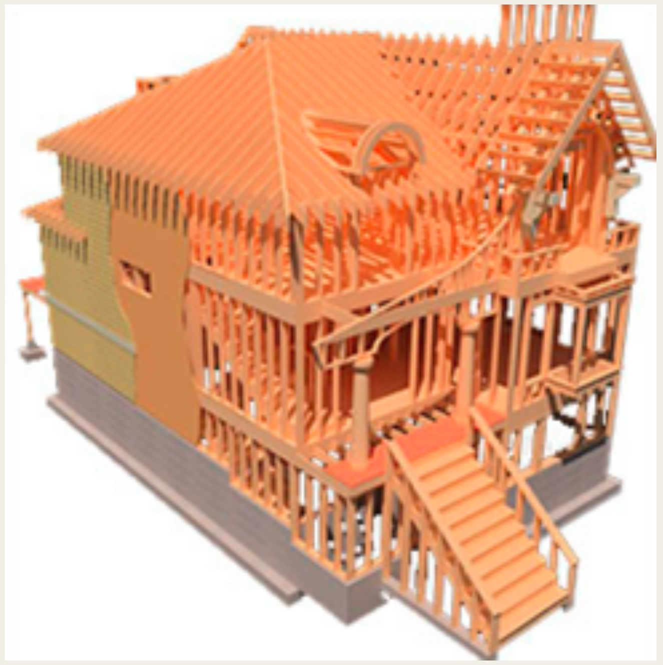
Lesson:

- As in software development, there are many things that could go wrong when building a tree out of 1000+ beer bottles…
- There are good processes to follow and there are bad processes to follow.
- Use a good process.

Other Metaphors?

- Software Farming: Growing a System
- Software Oyster Farming: System Accretion
- Software Construction: Building Software
- Software Penmanship: Writing Code

McConnell's Best Metaphor: Construction



- Incremental.
- Various stages: planning, execution, inspection.
- In house construction, materials are expensive, but major cost is the labor.. you want the design to be as good as possible so you don't waste time fixing mistakes.
- Similar build vs. buy tradeoffs.
- Different design models for different projects, e.g. house vs. power plant.
- Similarly, extremely large projects have different requirements than smaller ones.

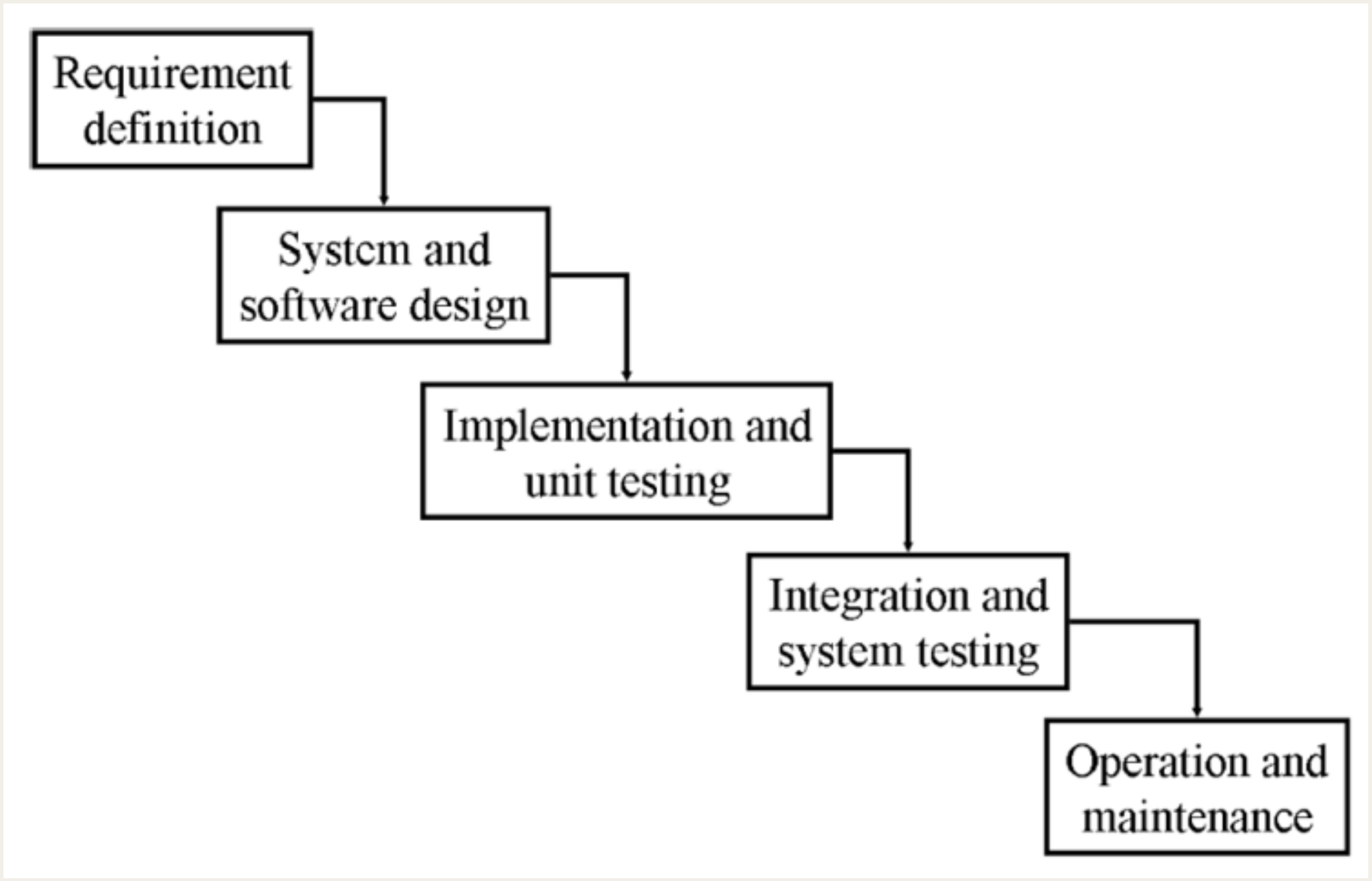
Ok, More Formally, What is a Software Development Process?

- A structured set of activities required to develop a software system.
  - Specification
  - Design
  - Implementation
  - Validation
  - Evolution
- Activities vary depending on the organization and the type of system being developed.

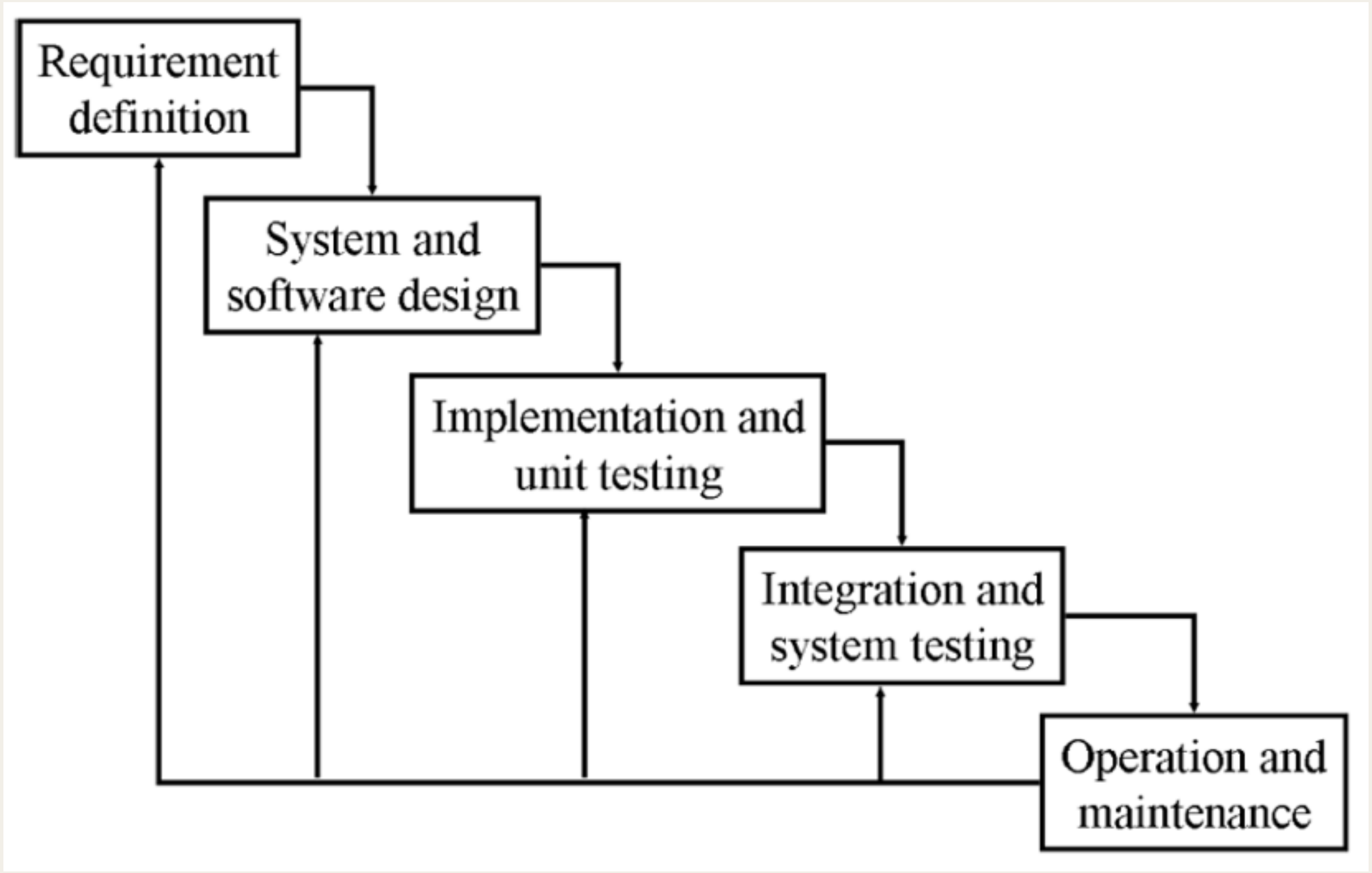
Software Process Models

- Waterfall Model
  - Breaks down a project based on activities: requirements analysis, design, coding, and testing.
  - A 1-year project might have a 2-month analysis phase, 4-month design phase, 3-month coding phase, and 3-month testing phase.
- Incremental / Iterative Development Model
  - Breaks down a project by subsets of functionality.
  - A 1-year project might be broken down into 3-month iterations. In the first, you take a quarter of the requirements and do the complete software life cycle for that 1st quarter: analysis, design, code, and test. Then, you have a complete system that works for a quarter of the needed functionality. In the next iteration, add the 2nd quarter of the functionality, etc..

The Waterfall Model


Requirement definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

The Waterfall Model with Feedback


Requirement definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Limitations of the Waterfall Method

- It’s very difficult to tell if the project is really on track.
- The difficulty of accommodating change once the process is underway.
- You seldom know the requirements that early.
- Difficult to evaluate the risk of the project and/or identify the most risky aspects of the project.
- Testing and integration are the hardest activities to estimate, so it’s difficult to support these at the end of the process.

More on the Waterfall Method

- The waterfall model is still the most widely used deliverable-based model.
- Best applied to:
  - projects where the requirements are very well known, e.g. the team has experience in the particular domain.
  - projects that are low risk.

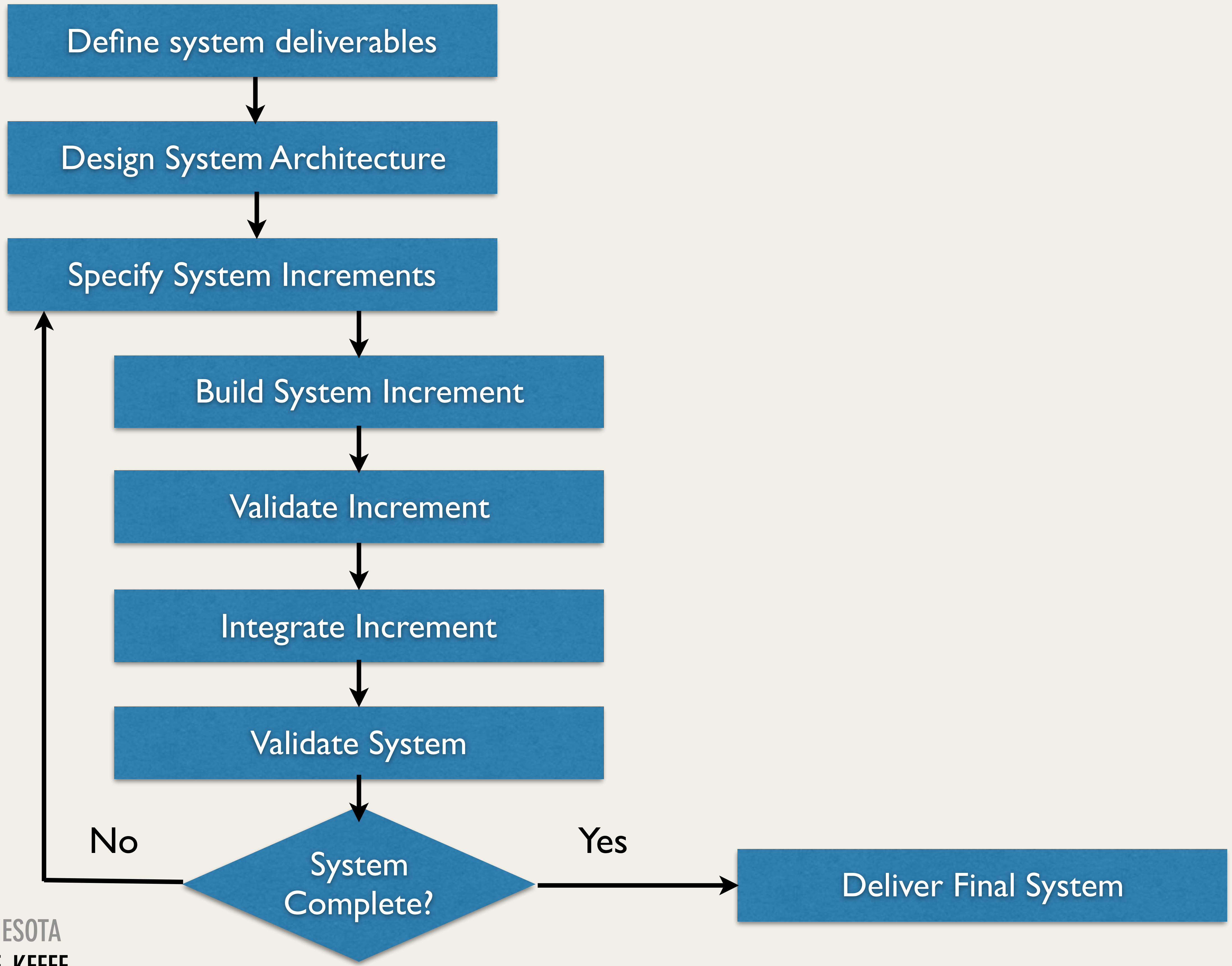
Iterative Development

- Is this iterative development (Fowler)?
  - “We are doing one analysis iteration followed by two design iterations...”
  - “The first iteration includes some bugs in the code that will be cleaned up in the second iteration.”
- No!

Iterative Development

- System is developed and delivered in increments after establishing an overall architecture.
- Users may experiment with delivered increments while others are being developed.
- These serve as a form of prototype system.

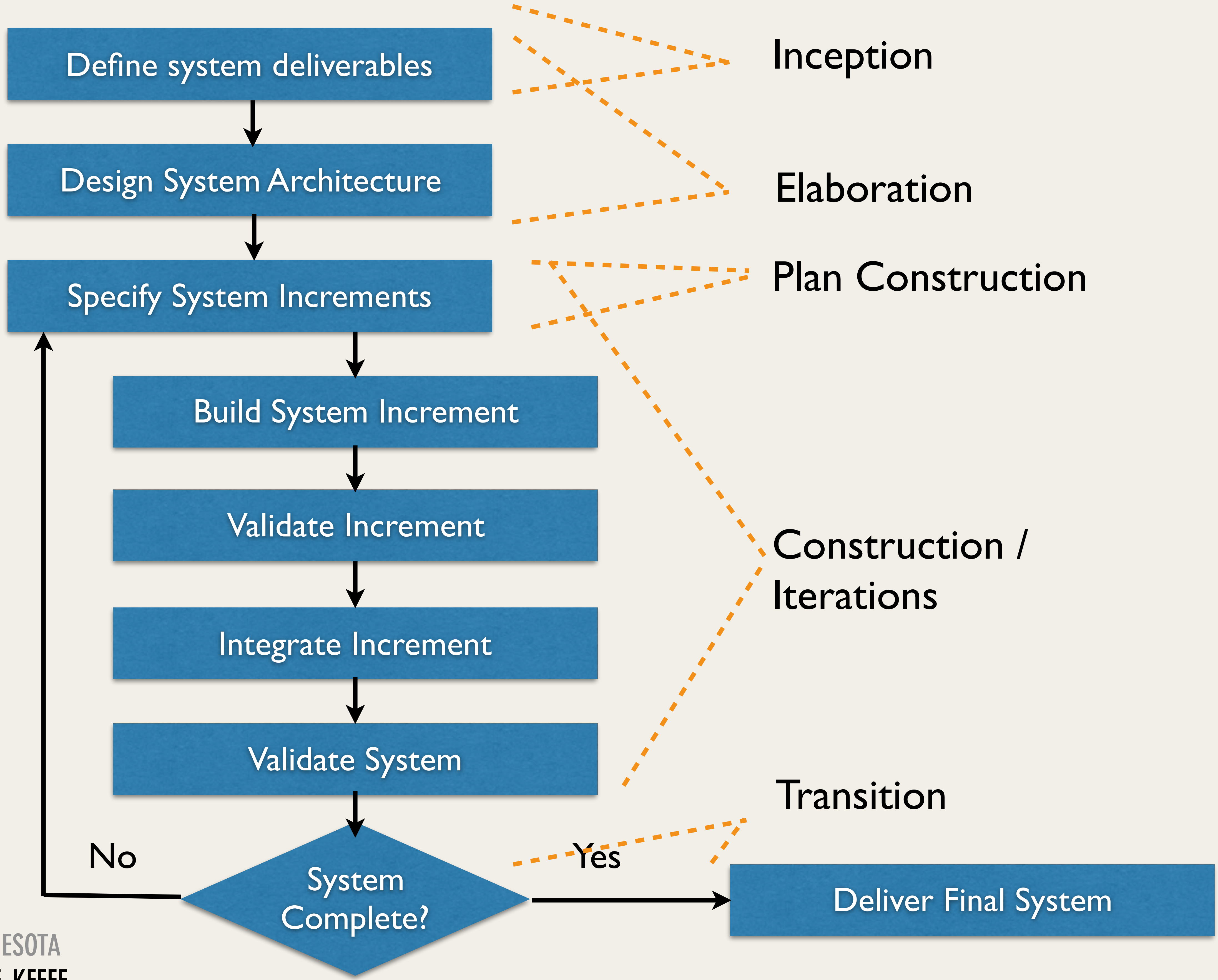
Iterative Development



Iterative Development -- Process Overview

- **Inception**
  - creation of the basic idea we want to implement, could be via discussion, could be a full fledged feasibility study.
  - outcome: project scope and business case.
- **Elaboration**
  - What is it you are going to build?
  - How are you going to build it?
  - What technology are you going to use?
  - Risk assessment
- **Plan Construction Iterations**
  - categorize use cases, “I must have this function”, “This is important, but I can live without it”, etc.
  - make time estimates and allocate the use cases to iterations
- **Construction**
  - each iteration is a “mini” project, a single iteration can follow the waterfall model: analyze, design, code, test, integrate.
- **Transition**: the phase between the beta release and the final product
  - performance evaluation and optimization
  - no new functionality, fix bugs.

Iterative Development



After Each Iteration:

- Code should be of near production quality.
- Should not have comments like, “This iteration’s code is buggy, but we’ll clean it up in the next iteration.”
- Many iterations are shown to the customer to get feedback -- called releases.

Iterative vs. Waterfall

- Waterfall skeptics point out it leaves two difficult and hard to predict activities to the end: system integration and system testing.
- The iterative model spreads these out across the entire development process.
- If each iteration is near production quality, then integration and testing will have been done properly.

Time Boxing

- Commonly used in iterative development.
- Fixes the amount of time allowed for each iteration.
- If planned features can’t be included, push them to another iteration.
- Forced to choose between slipping functionality and slipping release date.

Rework in Iterative Development

- Integrating the latest increment into the system often involves changing existing code.
- Isn’t this wasteful?
- Some of this work can be reduced by:
    - Automated regression testing.
    - Refactoring tools -- semi-automatic tools can often help with changes that improve readability/organization while preserving the interface to a program.
    - Continuous integration -- nightly builds.

Our Course

- We're following an iterative model. (Sort of.)