

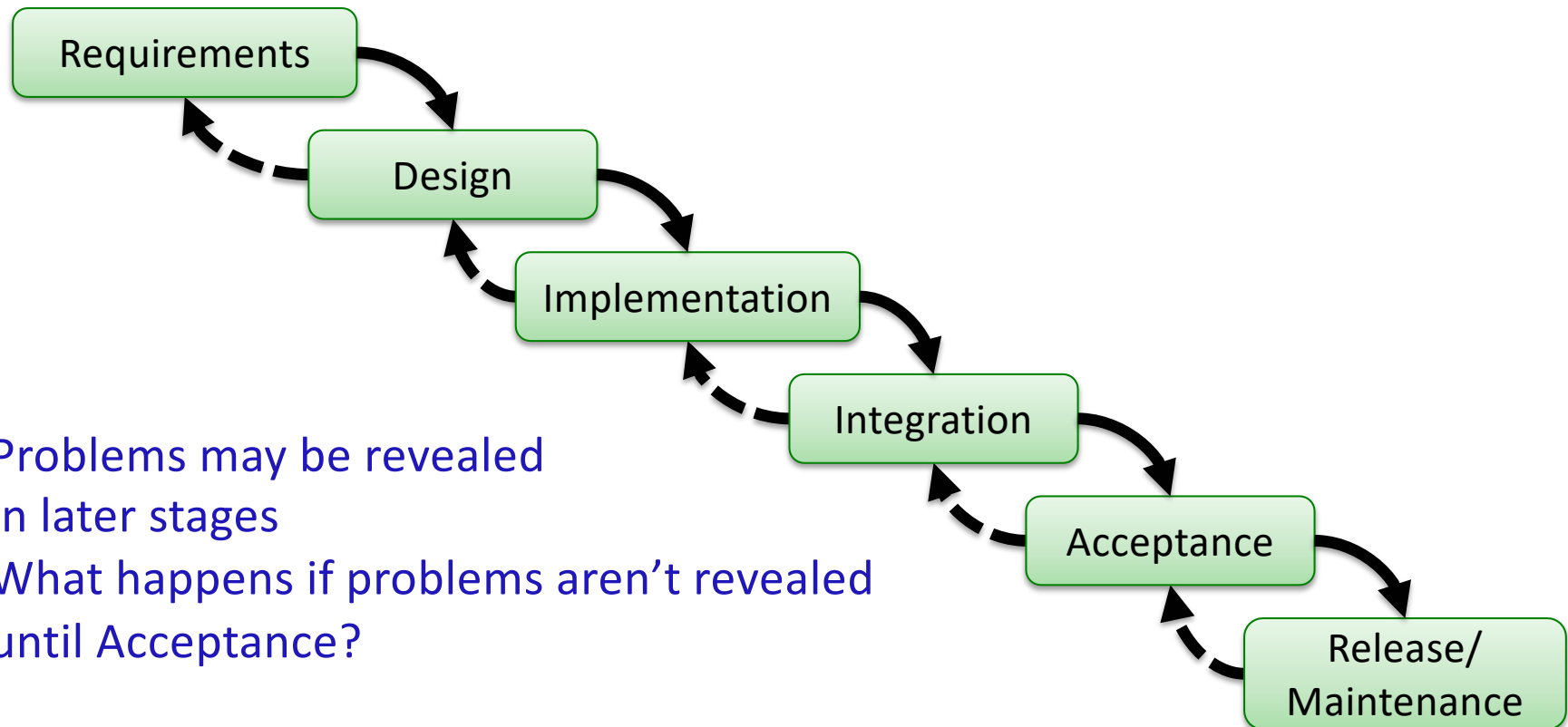
# Objectives

- Software Development
- Testing

# Review: Software Development

- What models for software development did we discuss?
- Describe the general testing process (from earlier this term)

# Feedback in Waterfall Model

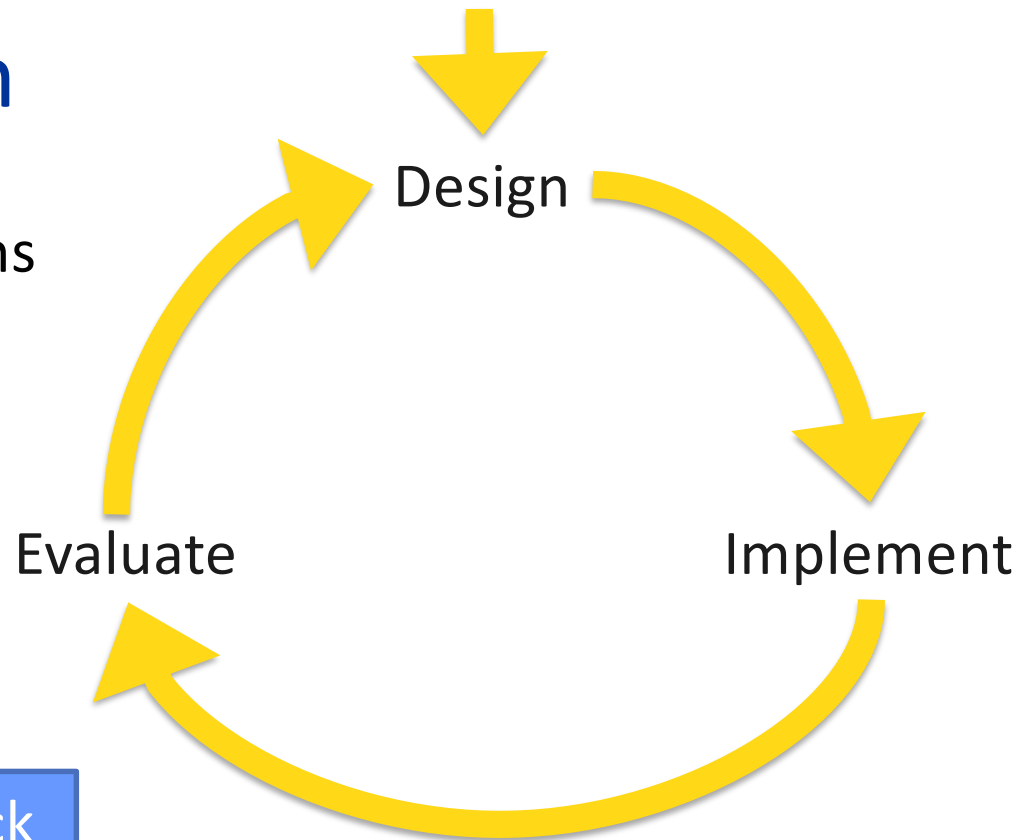


- Problems may be revealed in later stages
- What happens if problems aren't revealed until Acceptance?

# Iterative Design

Various implementations

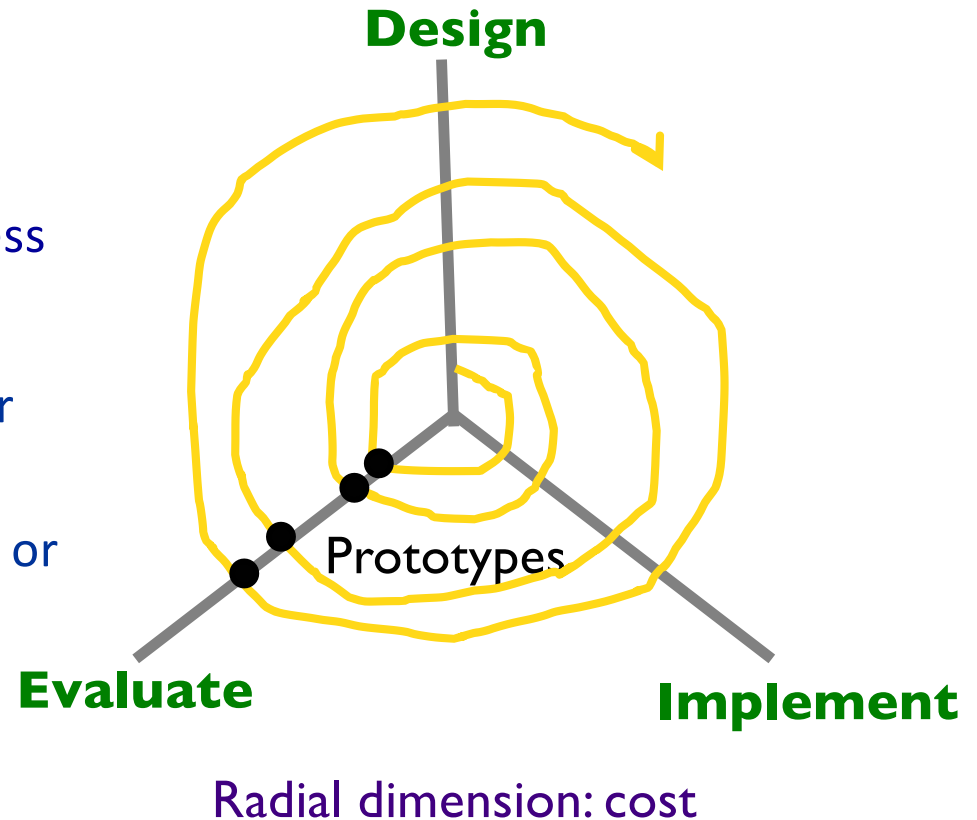
Get feedback/requirements  
from users/clients



Goals: Frequent feedback  
→ Identify problems early  
→ Higher quality product

# Spiral Model

- Idea: smaller prototypes to test/fix/throw away
  - Finding problems early costs less
- In general...
  - Break functionality into smaller pieces
  - Implement most depended-on or highest-priority features first

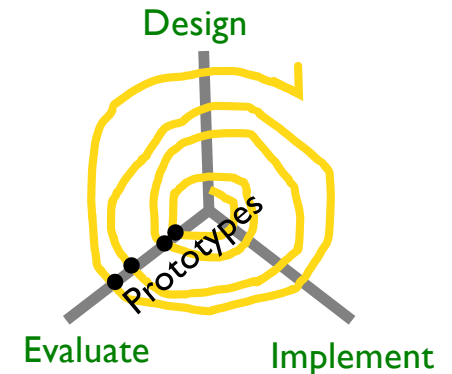


# Prototypes, In Brief

- Sample of application
  - Often: Demonstrate one part/purpose
  - Focus on one thing, not the whole thing
- Purpose/Dimensions
  - Functionality
  - Interaction
  - Implementation
- Fidelity
  - Low: omits details
  - High: very similar to finished product

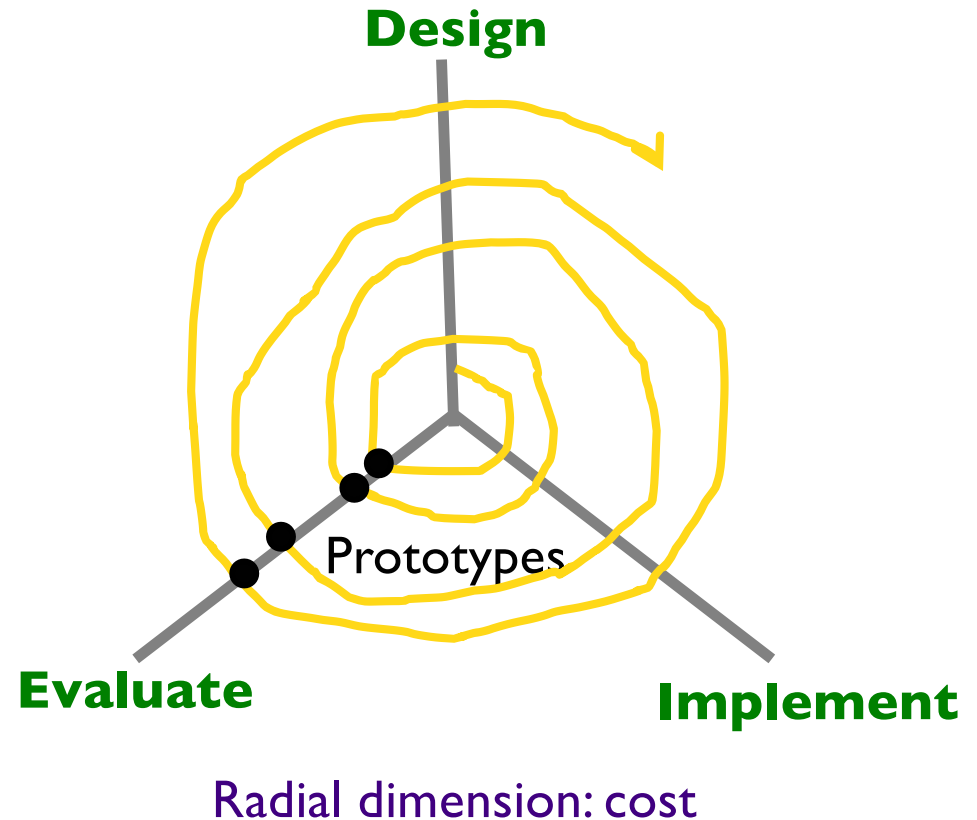
# Spiral Model/Iterative Design Model Benefits

- Builds in getting feedback from client
  - Demo prototypes or working versions of [parts of] application
  - Clients' requirements may change
  - Clients' requirements may be ambiguous or were misinterpreted
- Makes project development more **agile**
  - Goal: find problems early
  - Easier to throw away cheaper early prototypes
  - Adjust/adapt to changes



# Spiral Model: Breaking Down Further

- Project's development process: Spiral Model
- What does this look like day to day?
  - Agile development is a common implementation





# Agile Development

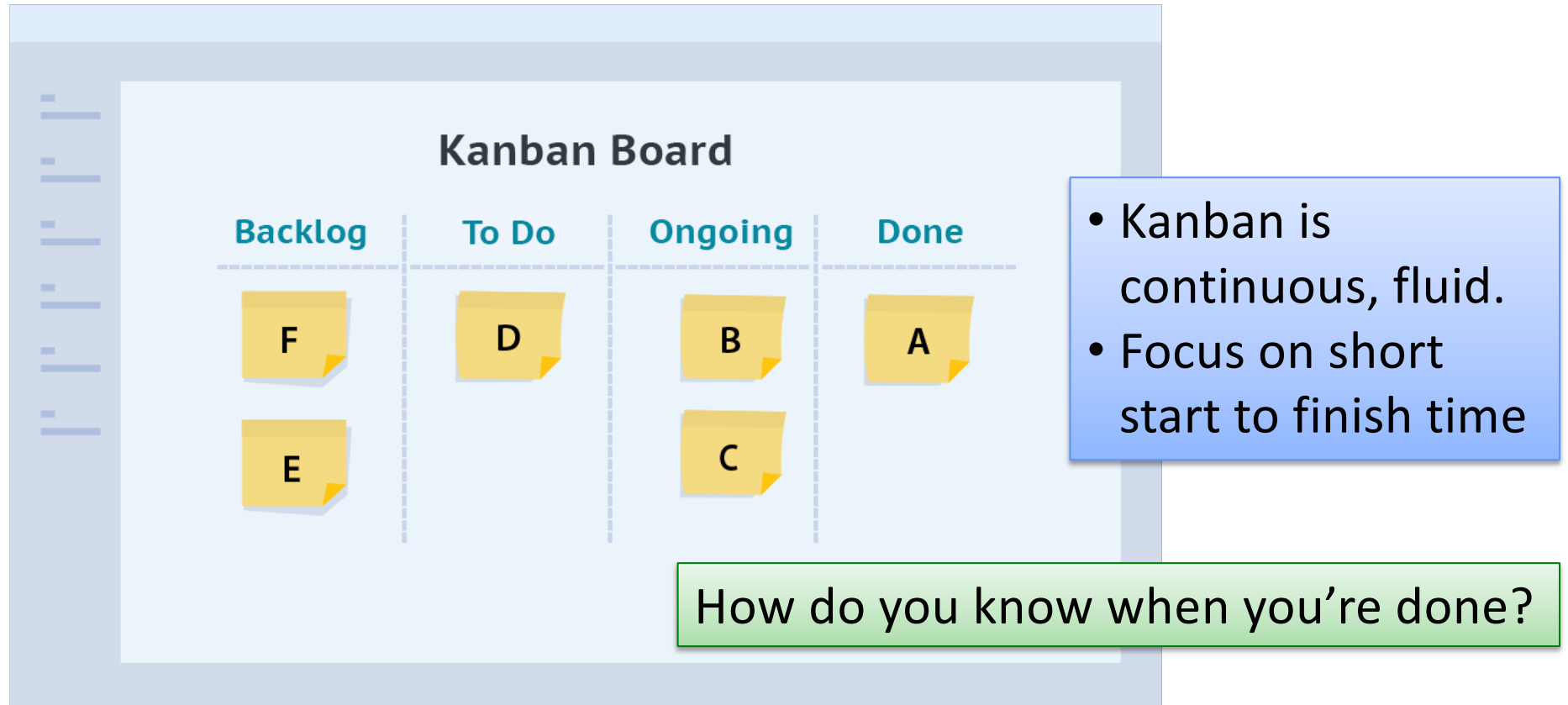
- Iterative approach to project management and software development
  - Work in small, launchable increments
  - Frequent review of requirements, plans, results
- Goals:
  - Respond to change quickly
  - Deliver application faster
  - Fewer conflicts about requirements
- Lots of variations – often company- or team-specific

# Agile Development Framework: Scrum

- Product owner creates prioritized wish list: a *product backlog*
- Team works in a *sprint*, usually 2-4 weeks
  - During planning, team picks a subset of wish list, a *sprint backlog*, and decides how to implement those pieces
  - Daily Scrum: team meets daily to assess its progress
    - ScrumMaster keeps the team focused on its goal
  - At end of sprint, work should be potentially shippable:
    - ready to hand to a customer, put on a store shelf, or show to a stakeholder
  - The sprint ends with a sprint review and retrospective
- Repeat sprint

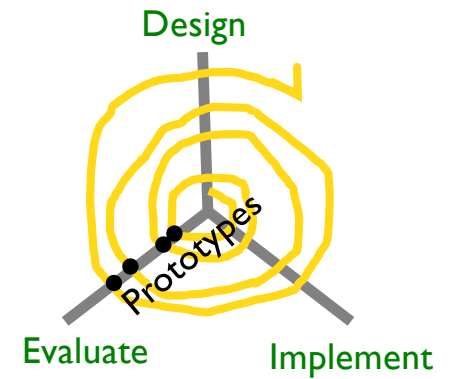
<https://www.scrumalliance.org/why-scrum>

# Tools to Help: Kanban Board



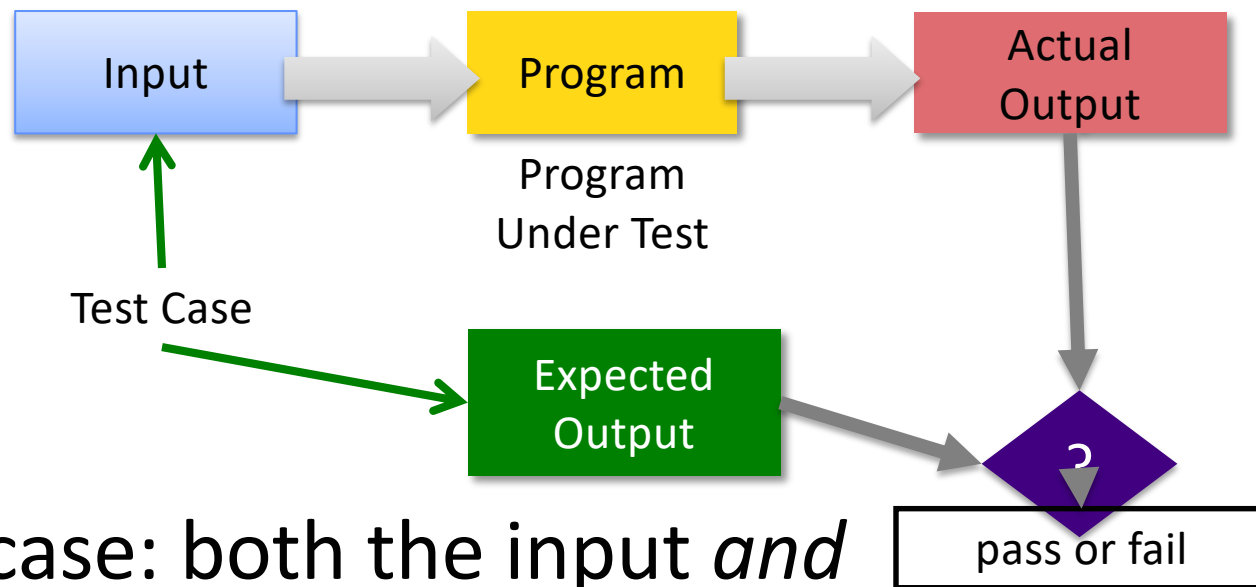
# Implementation Step

- Implementation includes developing and testing
- How do you know when you're done developing/testing?
  - Helpful to have a systematic way to know that you're done ✓



# SOFTWARE TESTING PROCESS

# Review: Software Testing Process



- Test case: both the input *and* the expected output
- Test Suite: set of test cases

# Type 1 Bugs: Compile-Time

- Syntax errors
  - Missing semicolon, parentheses
- Compiler notifies of error
- Cheap, easy to fix



## Type 2 Bugs: Run-Time

- Usually logic errors
- Expensive to locate, fix





# Aside: Objections to “Bug” Terminology

- “Bug”
  - Sounds like it’s just an annoyance
    - Can simply swat away
  - Minimizes potential problems
  - Hides programmer’s responsibility
- Alternative terms
  - **Defect**
  - **Fault**



# Tenor of Conversation

How do we detect bugs and fix them before the user sees them?

- NOT: how do we *never* write bugs?
  - We're human!
  - I mean, don't *try* to write bugs/be sloppy...
  - There's a balance.

# Discussion: Your Testing Process

- How do you test?
- Categorize what you test/look for
- Are you a good tester? Why or why not?
  - What do you do well?
  - What do you need to get better at?

# Common Bad Development Approaches

- Run the code. Did it do what you expect? <shrug/>
- Identify bug. Fix the bug on the test case that revealed the error. Don't test the other cases.
  - Similar: made a change to code (famous last words: "it shouldn't affect anything") and don't retest
- Tests don't help you identify the problem
  - A good set of tests will help you narrow the scope of the problem
- Random (only) testing

# Microsoft Windows Vista™ Testing

- Beyond their internal testing ...
  - 5 million people beta tested
  - 60+ years of performance testing
  - 1 Billion+ Office 2007 sessions
- Still, users found correctness, stability, robustness, and security bugs

# OSS Fuzz Project

- Continuous Fuzzing for Open Source Software
  - Fuzzing is a testing technique
- “Google has found thousands of security vulnerabilities and stability bugs by deploying guided in-process fuzzing of Chrome components”
- Also found 36K+ bugs in 1K+ projects

<https://github.com/google/oss-fuzz>

# Conclusion: Software Testing is Hard!

- Need to use a lot of different approaches
  - Different approaches catch different defects

# Types of Testing

(Non-Exhaustive)

- Black-box testing
- White-box testing
- Non-functional testing
- Acceptance testing

Ideas about or definitions of any of these?  
What is the approach? Or, what problems are they trying to reveal?



# Types of Testing

(Non-Exhaustive)

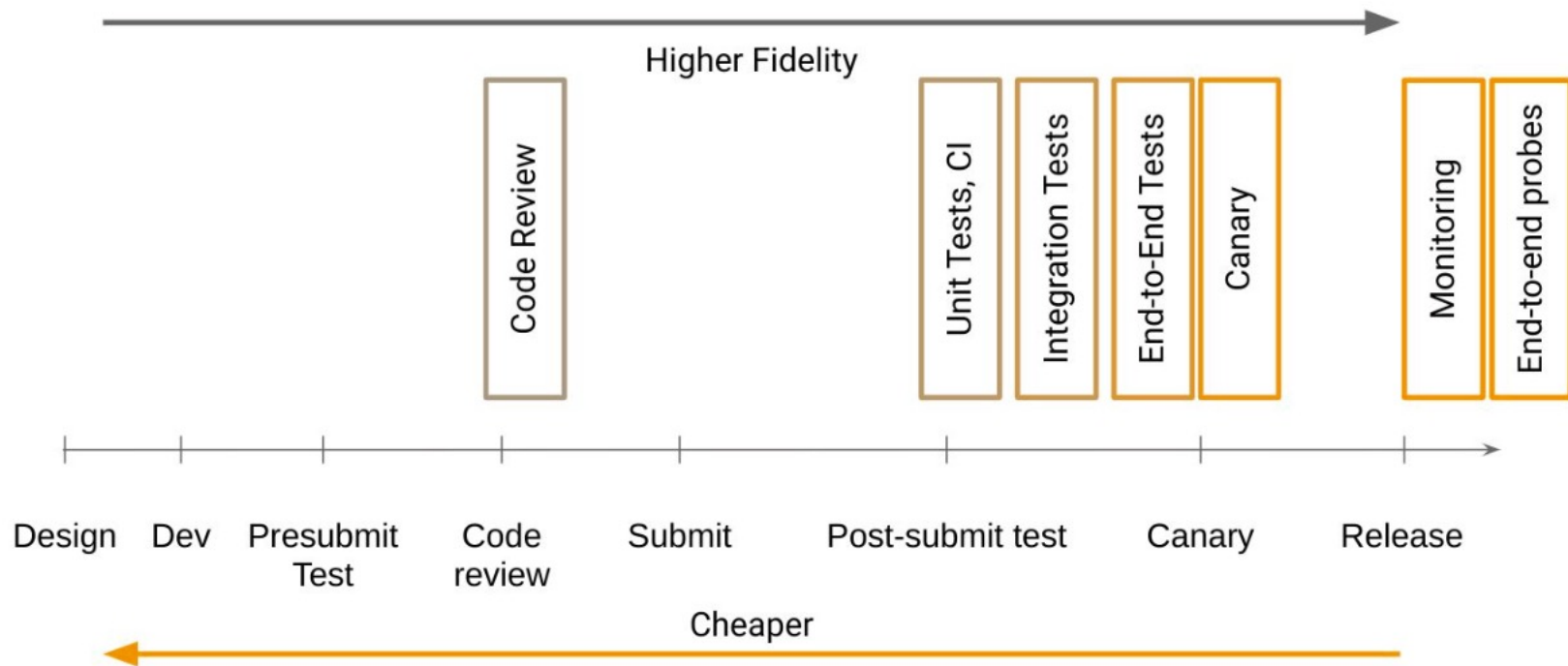
- Black-box testing
  - Test *functionality* (e.g., the calculator)
  - No knowledge of the code
  - Examples of testing: boundary values
- White-box testing
  - Have access to code
  - **Goal:** execute *all* code
- Non-functional testing
  - Performance testing
  - Usability testing (HCI)
  - Security testing
  - Internationalization, localization
- Acceptance testing
  - Customer tests to decide if they accept the product

# Levels of Functional Testing



- Unit
  - Tests minimal software component, in isolation
  - For us, Class-level testing
  - Web: Web pages (Http Request)
- Integration
  - Tests interfaces & interaction of classes
- System
  - Tests that completely integrated system meets requirements
- System Integration
  - Test system works with other systems, e.g., third-party systems

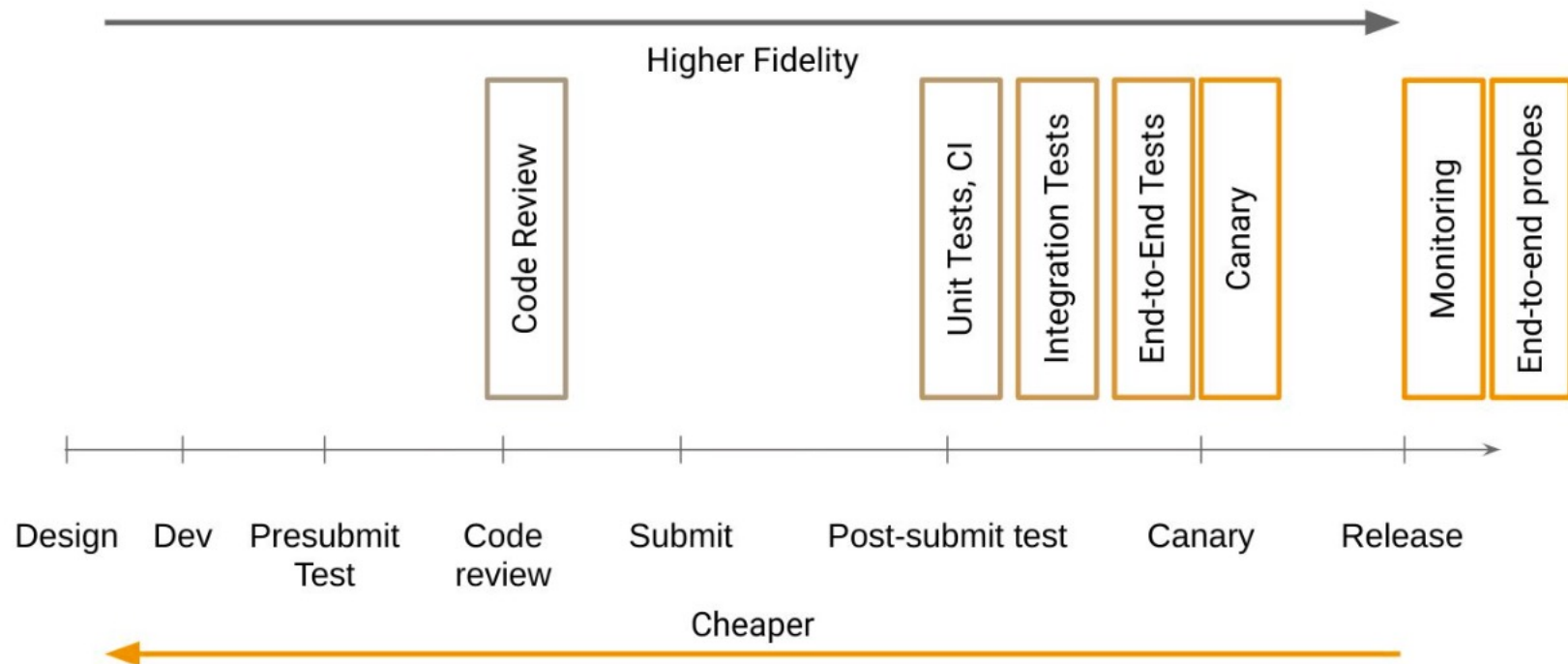
# Software Development Process



# A Bad Role Model



# Software Development Process



**Shift Left's Goal: Catch defects earlier**

## For Wednesday

- Read slides about JUnit testing
- Take reading quiz
- Set up project in Eclipse for in-class exercise