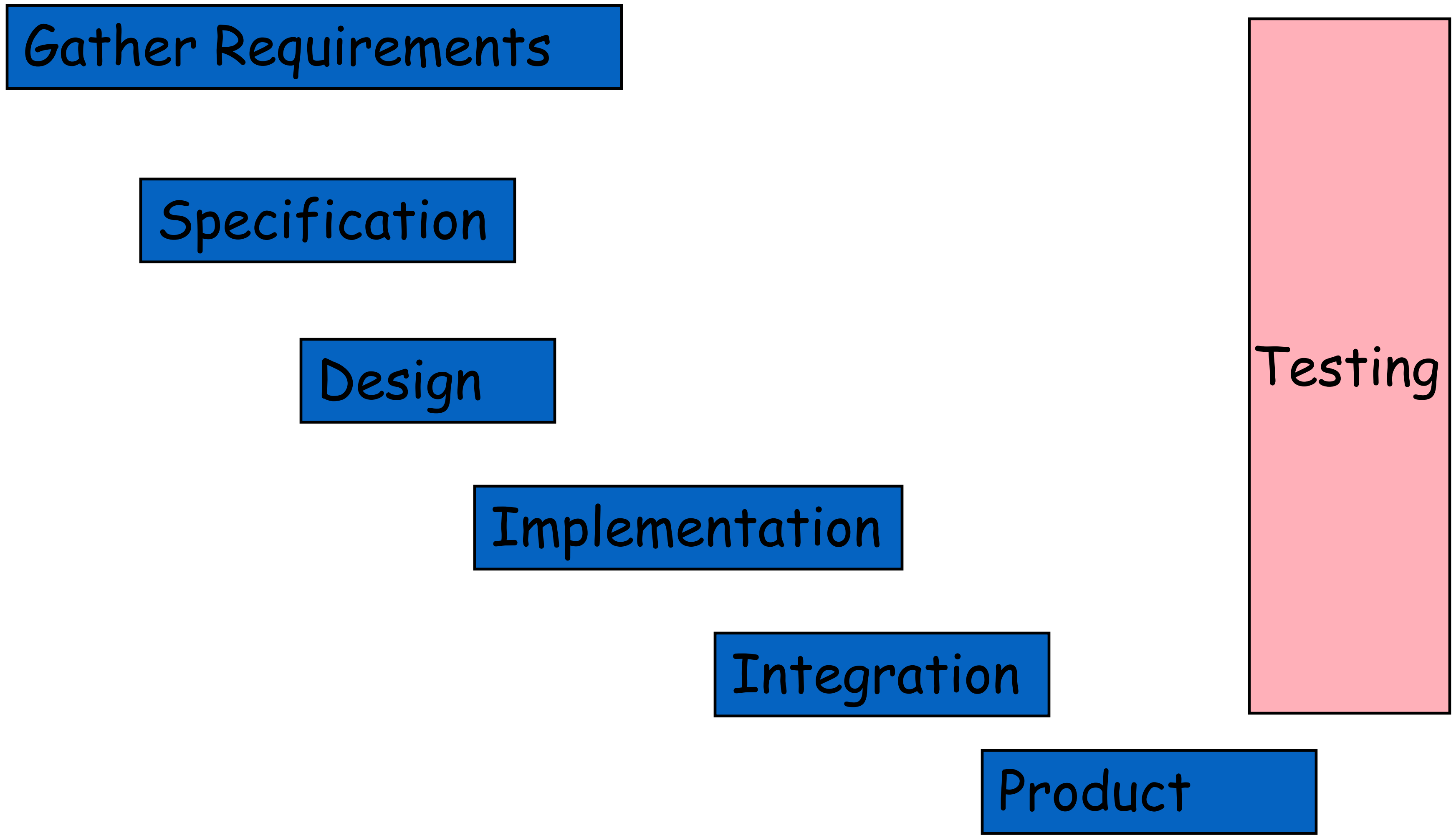
Software Development Processes: *Waterfall Model*

Software Engineering

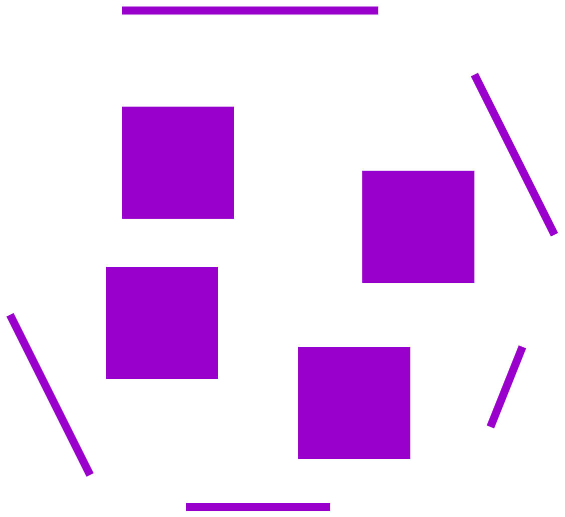
Millersville University

Waterfall Process Phases



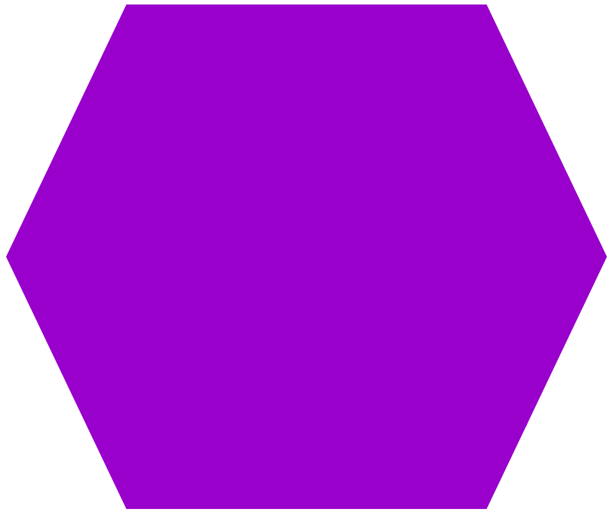
1. Gather Requirements

- Figure out what this thing is supposed to do
  - A raw list of features
  - Written down . . .
- Usually a good idea to talk to users, clients, or customers!
  - But note, they don't always know what they want
- Purpose:
  - Make sure we don't build the wrong thing
  - Gather information for planning



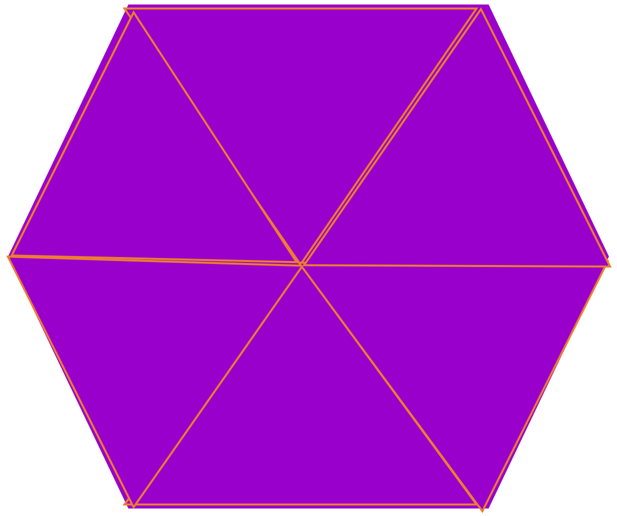
2. Specification

- A written description of *what* the system does
  - In all circumstances
    - For all inputs
    - In each possible state
- A written document
- Because it covers all situations, much more comprehensive than requirements



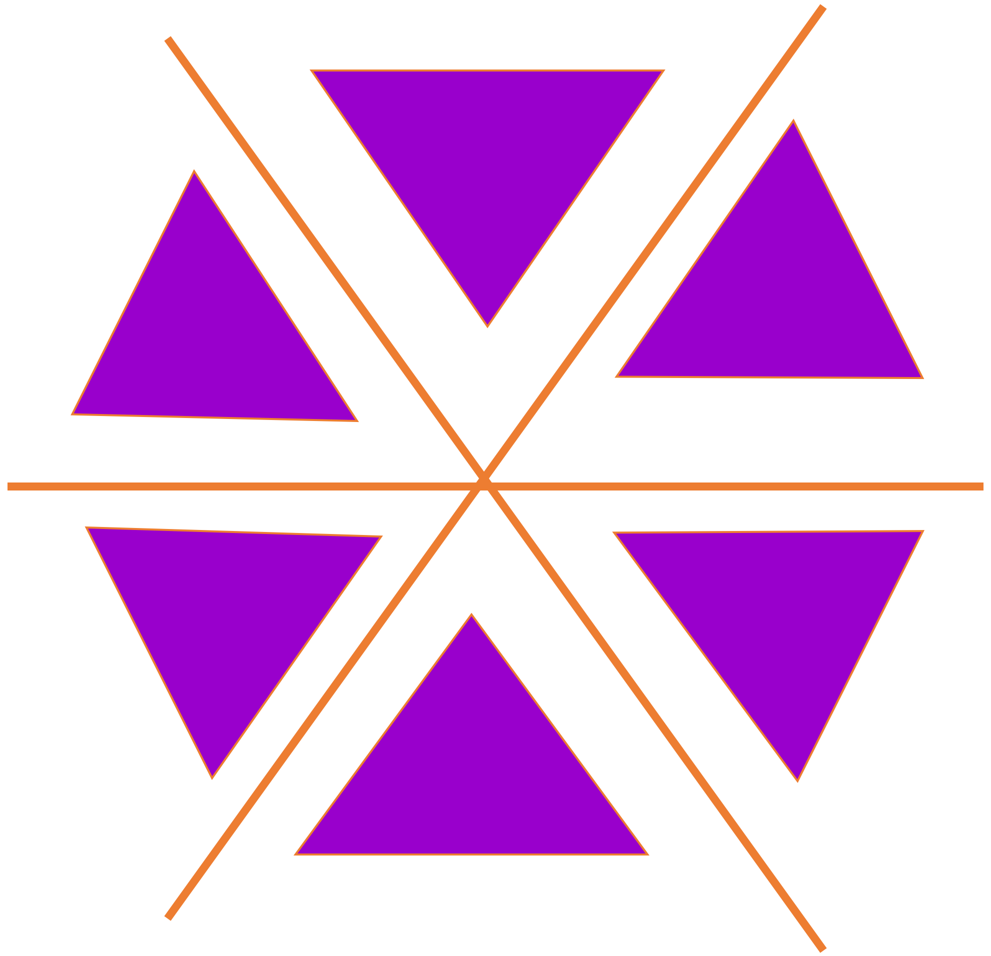
3. Design

- The system architecture
- Decompose system into modules
- Specify interfaces between modules
- Much more of *how* the system works, rather than *what* it does



3. Design

- The system architecture
- Decompose system in modules
- Specify interfaces between modules
- Much more of *how* the system works, rather than *what* it does

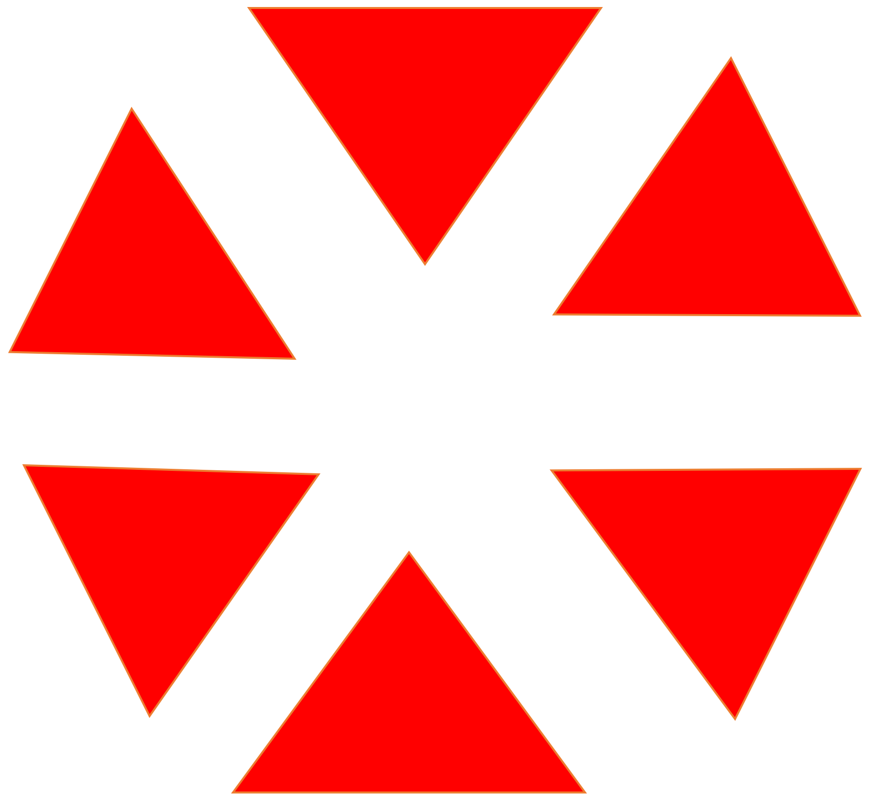


4. Implementation

- Code up the design

- First, make a plan
  - The order in which things will be done
  - Usually by priority
  - Also for testability

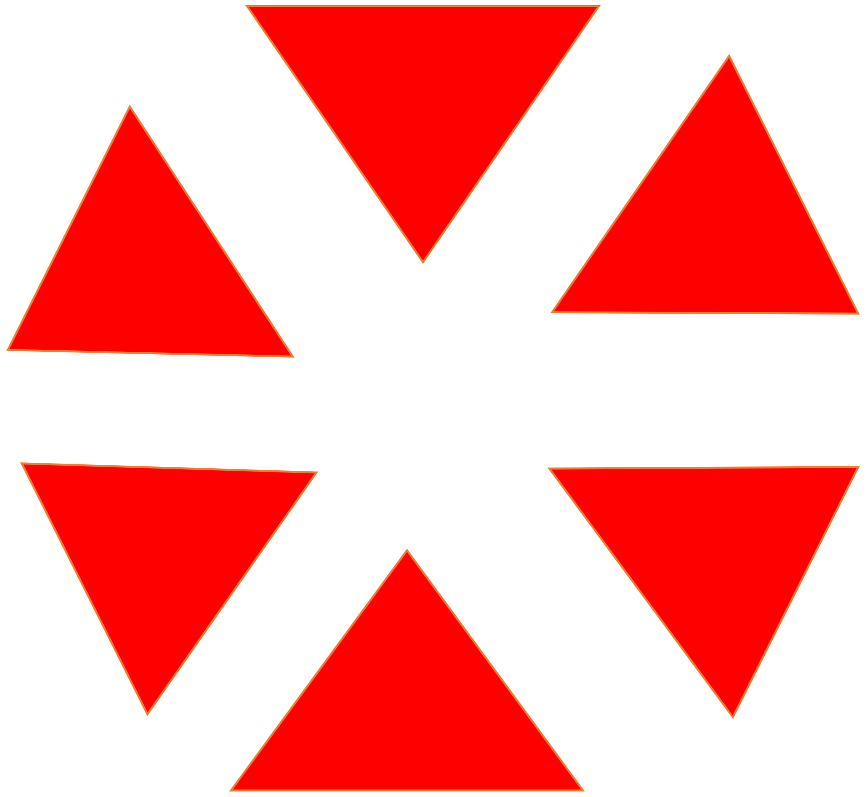
- Test each module



5. Integration

- Put the pieces together

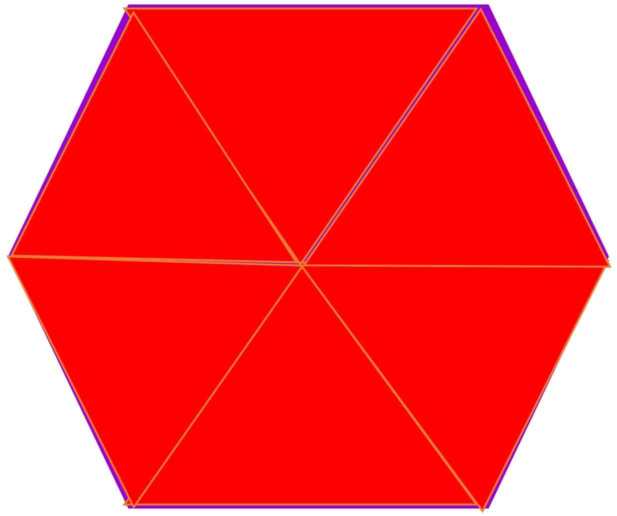
- A major QA effort at this point to test the entire system



5. Integration

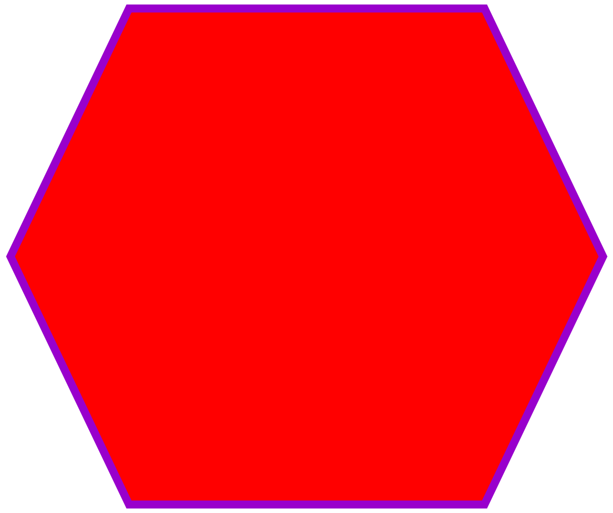
- Put the pieces together

- A major QA effort at this point to test the entire system



6. Product

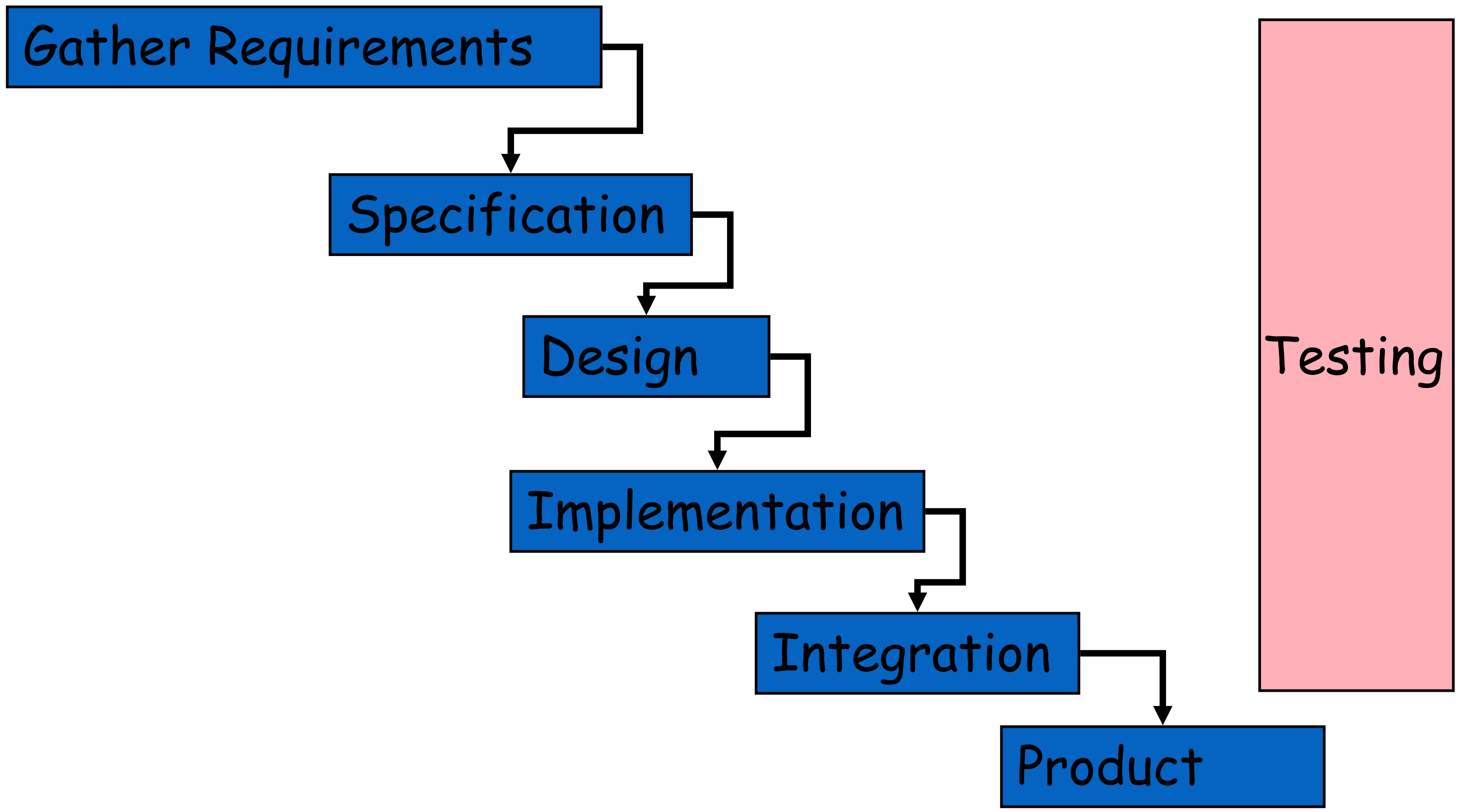
- Ship and be happy
- Actually, start maintenance



A Software Process: Waterfall Model

- One of the standard models for developing software
- Each stage leads on to the next
    - No iteration or feedback between stages

The Waterfall Model



The Waterfall Model (Cont.)

- There is testing after each phase
    - Verify the requirements, the spec, the design
    - Not just the coding and the integration

- Note the top-down design
    - Requirements, spec, design

- Bottom-up implementation
    - Implement, integrate subparts, integrate product

The Waterfall Model (Discussion)

- What are the risks with the waterfall model?

Opinions

- The major risks are:
  - Relies heavily on being able to accurately assess requirements at the start
  - Little feedback from users until very late
    - Unless they understand specification documents
  - Problems in the specification may be found very late
    - Coding or integration
  - Whole process can take a long time before the first working version is seen
    - Frequent intermediate builds are needed to build confidence for a team
  - Sequential
    - The programmers have nothing to do until the design is ready

Opinions

- The waterfall model seems to be adopted from other fields of engineering
    - This is how to build bridges
- I believe very little software is truly built using the waterfall process
    - Where is it most, least applicable?
- But many good aspects
    - Emphasis on spec, design, testing
    - Emphasis on communication through documents

An Opinion on Time

- Time is the enemy of all software projects

- Taking a long time is inherently risky

"It is hard to make predictions,
especially about the future"

Why Time is Important?

- The world changes, sometimes quickly
- Technologies become obsolete
  - Many products obsolete before they first ship!
- Other people produce competitive software
- Software usually depends on many 3rd-party pieces
  - Compilers, networking libraries, operating systems, etc.
  - All of these are in constant motion
  - Moving slowly means spending lots of energy keeping up with these changes

A Case Study

- California DMV software (‘87-’93)
- Attempt to merge driver & vehicle registration systems
  - thought to take 6 years and $8 million
- Spent 7 years and $50 million before pulling the plug
  - costs 6.5x initial estimate & expected delivery slipped to 1998 (or 11 years)!