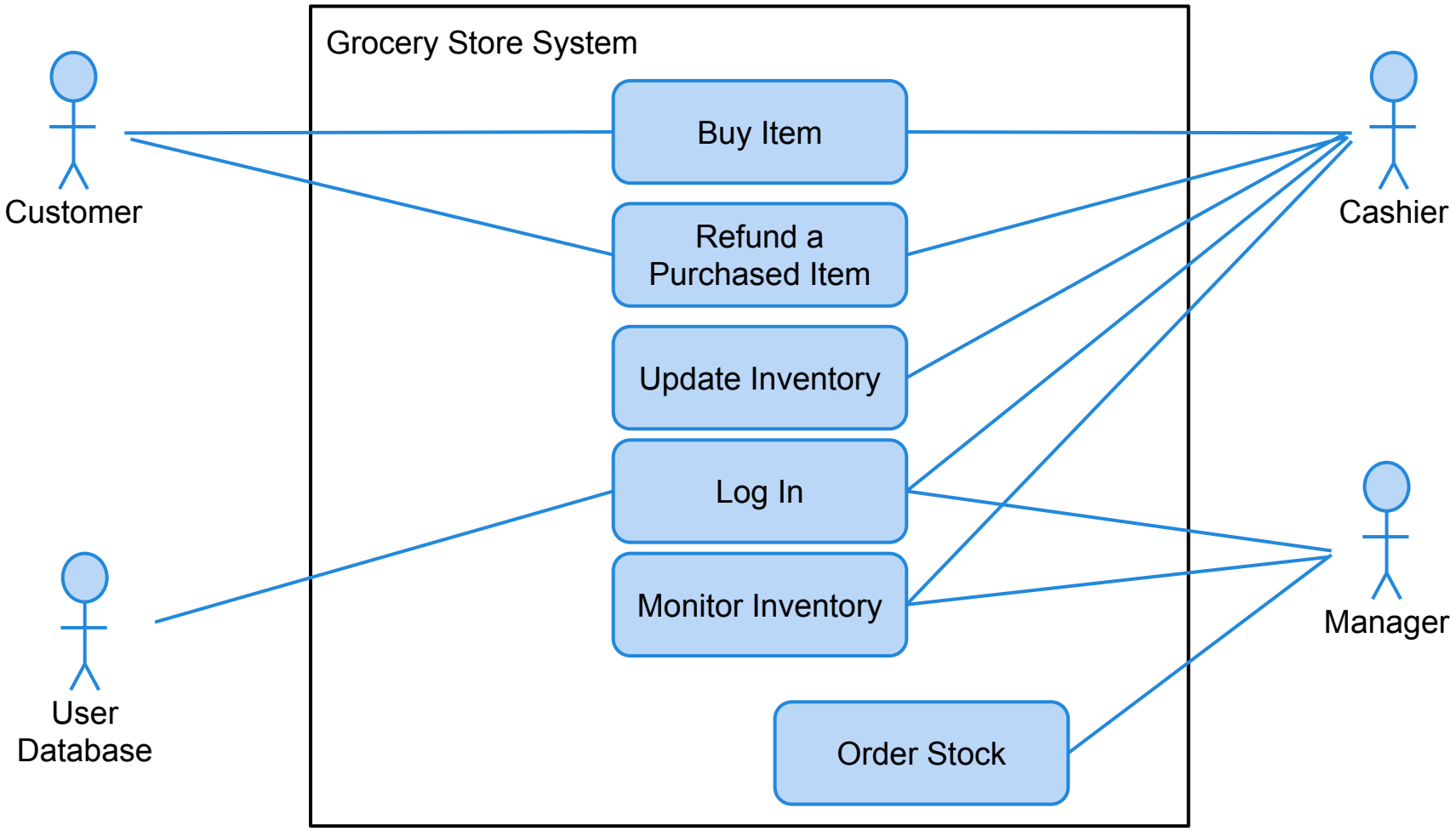
First…

A little more about use cases.

Grocery Store System Diagram



Grocery Store System Scenario

- **Scenario: Buy Item**
- **Actors:**
  - Customer (initiator), Cashier
- **Description:**
  - The Customer arrives at the checkout with items to purchase.
  - For each item, the Cashier records the item and the software updates the payment total.
  - The Cashier accepts payment in either cash or credit card form and records payment information in the software.
  - If payment is successful, the software will print a receipt and the Customer collects the items and leaves the store.

Grocery Store System Use Case

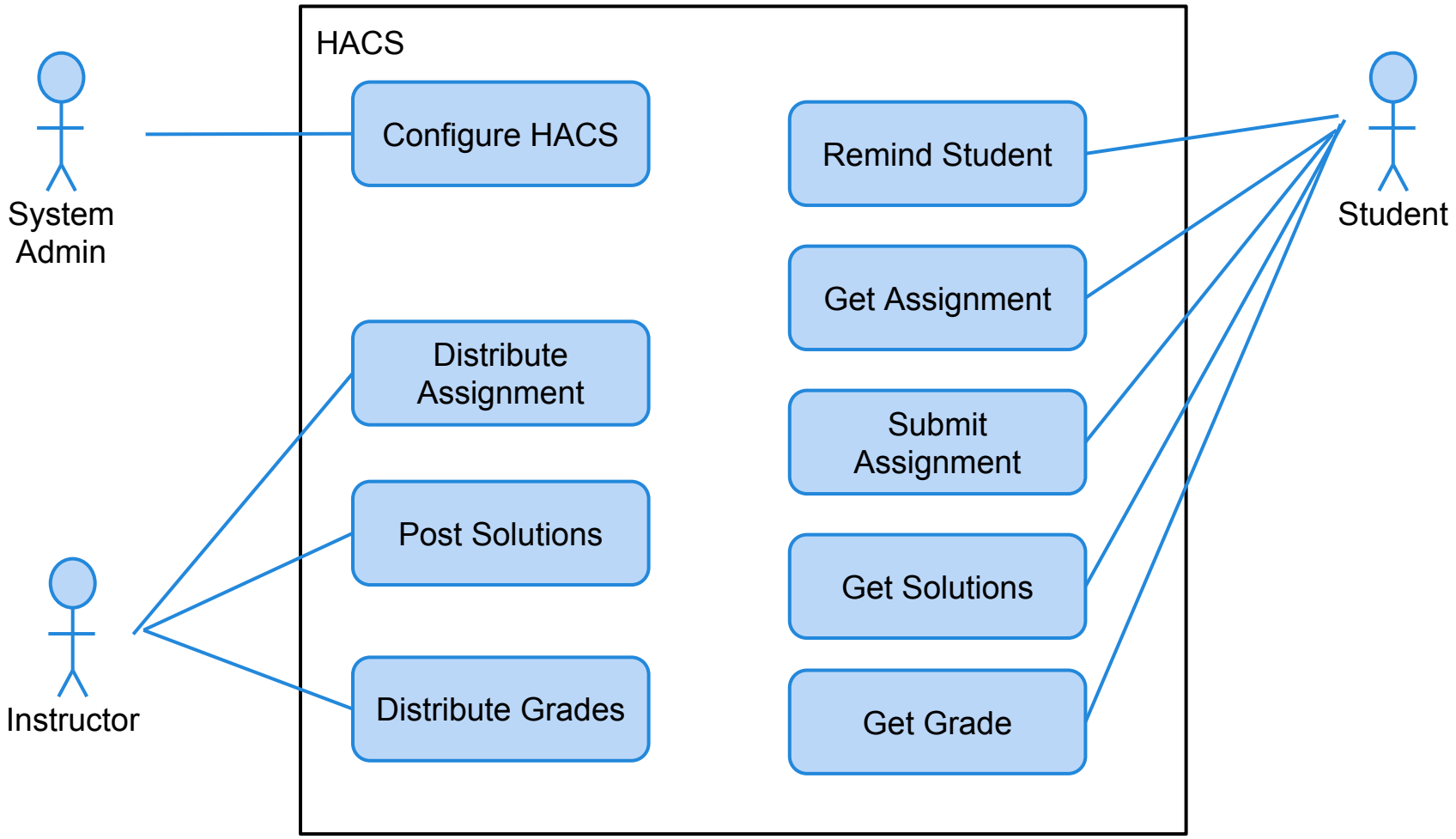
- **Use-Case: Buy Item**
- **Actors:** Customer (initiator), Cashier
- **Description:**
  - The Customer arrives at the checkout with items to purchase.
  - For each item:
    - the Cashier records the item,
    - completes use-case “Update Inventory”,
    - and the software updates the payment total.
  - The Cashier accepts payment in either cash or credit card form and records payment information in the software.
  - If payment is successful, the software will print a receipt and the Customer collects the items and leaves the store.
- **Exception Paths:** If credit card payment is denied, then an error message will be displayed and the customer will not be allowed to leave with the items.
- **Preconditions:** Cashier must have completed use-case “Log In”

Activity: HACS

Homework assignment and collection are an integral part of any educational system. Today, this is performed manually. We want to automate this with the Homework Assignment and Collection System (HACS).

HACS will be used by the instructor to distribute the homework assignments, review the students' solutions, distribute suggested solutions, and distribute student grades on each assignment. HACS shall also help the students by automatically distributing the assignments to them, providing a facility where the students can submit their solutions, reminding the students when an assignment is almost due, and reminding the students when an assignment is overdue.

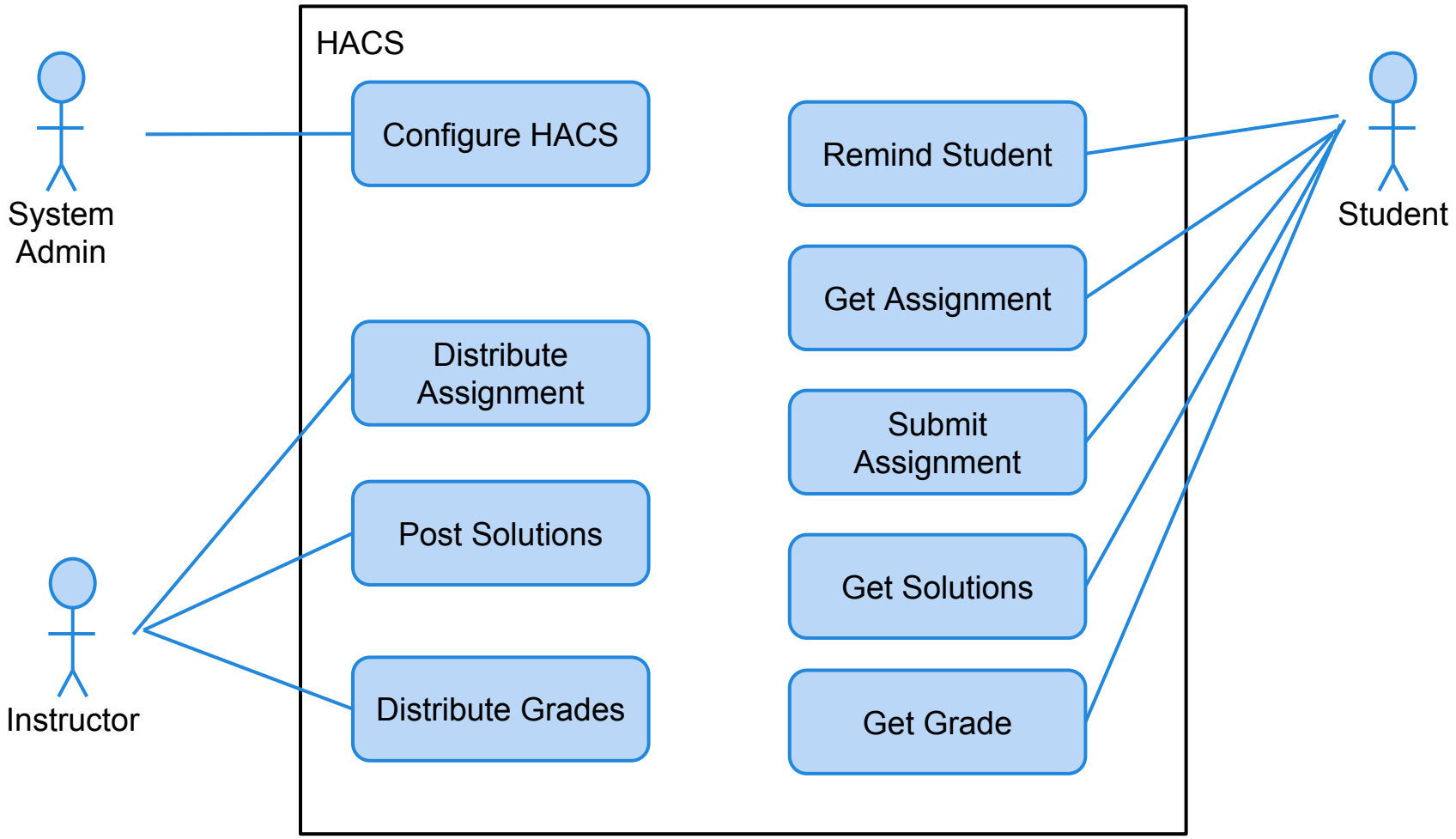
HACS Use Case Diagram



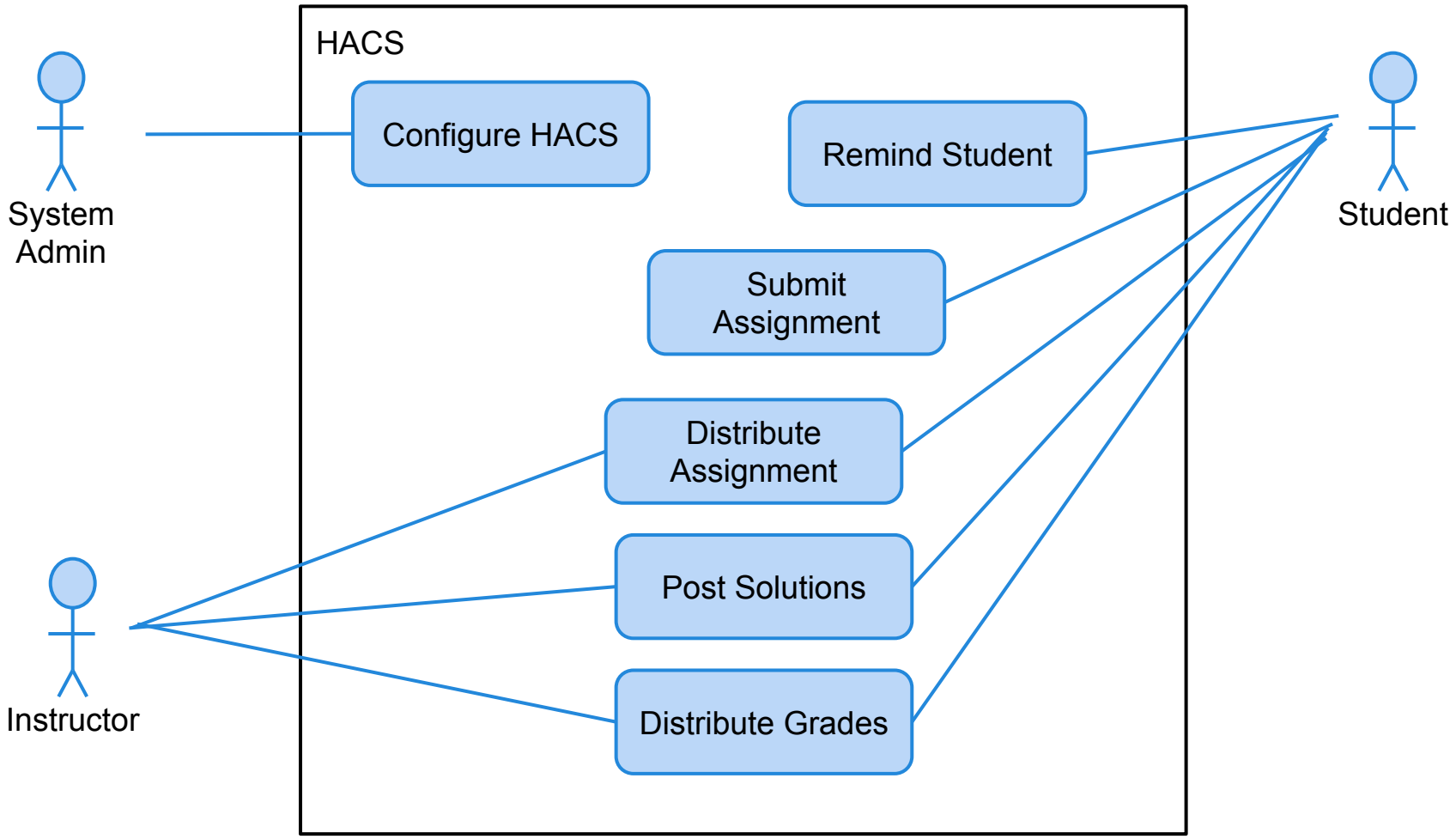
HACS Use Case: Distribute Assignment

- **Use-Case: Distribute Assignment**
- **Actors:** Instructor (initiator)
- **Description:**
  - The Instructor uploads an assignment to the system.
  - If the upload completes successfully, the Instructor will be asked to evaluate a preview of the file.
  - If the Instructor approves the file preview, HACS will ask for a due date.
  - Once the due date is submitted, the assignment will be added to the system and made readable for students, and the Instructor will be returned to the main menu.
- **Exception Paths:** If the file upload fails, an error message will be displayed, and the Instructor returned to the main menu.
- **Alternate Paths:** At any time, the Instructor may click the cancel button to return to the main menu.
- **Preconditions:** Use-Case “Configure HACS” must be performed before assignments can be distributed.

HACS Use Case Diagram



HACS Use Case Diagram (Version 2)



HACS Use Case: Distribute Assignment (Version 2)

- **Actors:** Instructor (initiator), Student
- **Description:**
  - The Instructor uploads an assignment to the system.
  - If the upload completes successfully, the Instructor will be asked to evaluate a preview of the file.
  - If the Instructor approves the file preview, HACS will ask for a due date.
  - Once the due date is submitted, the assignment will be added to the system and the Instructor will be returned to the main menu.
  - HACS will then make the assignment readable for students and e-mail each student a link to the file, along with a due date notice.
- **Exception Paths:** If the file upload fails, an error message will be displayed, and the Instructor returned to the main menu.
- **Alternate Paths:** At any time, the Instructor may click the cancel button to return to the main menu.
- **Preconditions:** Use Case “Configure HACS” must be performed before assignments can be distributed.

Things to Keep in Mind

- Remember:
  - Each use case will likely correspond to many requirements. Use cases are high level goals, requirements are low level statements of how to make that goal achievable.
  - Use cases represent an external view of the system. They do not tell you what your system objects are, and should not feature internal objects as actors.
  - No “rule of thumb” for how many use cases you should have:
    - Ask yourself: does this capture all of the goals a user might have when using my system?

Testing the Requirements

CSCE 740 - Lecture 7 - 09/16/2015

Today's Goals

- Discuss the importance of test cases for the requirements.
  - Help write better requirements
  - Verification and Validation
- How to come up with those test cases.

Requirements Verifiability

"The system should be easy to use by experienced engineers and should be organized in such a way that user errors are minimized."

- Problem is the use of vague terms such as "errors shall be minimized."
- The error rate must be quantified

Why Write Tests Based on the Requirements?

- The software might have bugs.
- *The requirements might have "bugs"*.
  - Can't automatically check this, but writing a test requires thinking through the requirement.
- Gives a way to argue that the software does what we promised it would do (**verification**).

Verification and Validation

Activities that must be performed to consider the software “done.”

- **Verification:** The process of proving that the software meets its stated functional and non-functional requirements.
- **Validation:** The process of proving that the software meets the customer’s needs and expectations.

Verification and Validation

Barry Boehm, inventor of “software engineering” describes them as:

- **Validation:** “Are we building the right product?”
- **Verification:** “Are we building the product right?”

Goal of V&V

The goal of V&V is to establish confidence that the system is “fit for purpose.”

How confident do you need to be? Depends on:

- **Software Purpose:** The more critical the software, the more important that it is reliable.
- **User Expectations:** When a new system is installed, how willing are users to tolerate bugs because benefits outweigh cost of failure recovery.
- **Marketing Environment:** Must take into account competing products - features and cost - and speed to market.

Definition of Software Testing

Investigation conducted to provide information about system quality.

Sequences of **stimuli** and **observations**.

$$(I_1 \rightarrow O_1) \rightarrow (I_2 \rightarrow O_2) \rightarrow (I_3 \rightarrow O_3)$$

What Does Testing Accomplish?

Your current goal shapes what scenarios the tests cover:

- **Defect Detection:** Discover situations where the behavior of the software is incorrect.
  - Tests tend to reflect extreme usage.
- **Verification:** Demonstrate to the customer that the software meets the requirements.
  - Tests tend to reflect “normal” usage.

Requirements-Based Testing

- Typically the baseline technique for designing test cases.
- Can begin as part of requirements specification, and continue through each level of design and implementation.
- Effective at finding some classes of faults that elude code-based techniques.
  - Namely - missing functionality

What Goes Into a Test?

- The anatomy of a test case
  - **Inputs** (test data) to the system.
  - Predicted **outputs** based on these inputs.
  - **Procedure** needed to exercise the system.
    - Pre-conditions and set-up steps.
    - Things that we will need to do to gather data.

Typical Requirements

- After a high temperature is detected, an alarm must be raised quickly.
- Novice users should be able to learn the interface with little training.

How in the world do you make these requirements verifiable?

Test the Requirement

After a high temperature is detected, an alarm must be raised quickly.

Test Case 1:

- Input:
  - Artificially raise the temperature above the high temperature threshold.
- Procedure:
  - Measure the time it takes for the alarm to come on.
- Expected Output:
  - The alarm shall be on within 2 seconds.

Test the Requirement

Novice users should be able to learn the interface with little training.

Test Case 2:

- Input:
  - Identify 10 new users and put them through the training course (maximum length of 6 hours)
- Procedure:
  - Monitor the work of the users for 10 days after the training has been completed
- Expected Output:
  - The average error rate over the 10 days shall be less than 3 entry errors per 8 hours of work.

“Fixed” Requirements

- **Original:** After a high temperature is detected, an alarm must be raised quickly.
- **New:** When the temperature rises over the threshold, the alarm must activate within 2 seconds.

- **Original:** Novice users should be able to learn the interface with little training.
- **New:** New users of the system shall make less than 2 entry mistakes per 8 hours of operation after 6 hours of training.

Detailed is Not Always Testable

- Number of invalid attempts to enter the PIN before a user is suspended.
  - This count is reset when a successful PIN entry is completed for the user.
  - The default is that the user will never be suspended.
  - The valid range is from 0 to 10 attempts.

Problem: “never” is not testable.

(same for “always”)

Patient Management System

Consider related requirements for a patient management system:

- If a patient is known to be allergic to any particular medication, then prescription of that medication shall result in a warning message being issued to the system user.
- If a prescriber chooses to ignore an allergy warning, they shall provide a reason why this has been ignored.

Patient Management System Tests

Some possible tests include:

- Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system.
- Set up a patient record with a known allergy. Prescribe the medication they are allergic to, and check that a warning is issued.
- Set up a patient record where allergies to two or more drugs are recorded. Prescribe both separately and check that the correct warning is issued for each.
- Prescribe both drugs at once and check that both warnings are issued.
- Prescribe a drug that issues a warning and overrule the warning. Check that the system requires the user to provide information explaining why the warning was overruled.

How Many Tests Do You Need?

Testing a requirement does not mean writing a single test.

- You normally have to write several tests to ensure that the requirement holds.
  - What are the different conditions that the requirement must hold under?
- Maintain traceability links from tests to the requirements they cover.

Scenario Testing

One method of testing is to use **scenarios** to develop test cases for the system.

- Stories that describe one way in which a system might be used.
  - Use-cases, user stories, sequences of user interactions.
- Stories should be complex and credible.
- Should be easy to evaluate.

Scenario Example

For the patient management system:

Kate is a nurse. One of her responsibilities is to visit patients at home to check on the progress of their treatment. On a day for home visits, Kate logs into the PMS and uses it to print her schedule of home visits for that day, along with summary information about the patients to be visited. She requests that the records for these patients be downloaded to her tablet. She is prompted for her password to encrypt the records for the tablet.

One of the patients, Jim, is being treated for depression. Jim feels that the medicine is keeping him awake at night. Kate looks up Jim's record and is prompted for her key phrase to decrypt the record. She checks the drug prescribed and queries its side effects. She notes the problem in Jim's record and enters a prompt to call him when she gets back to the office to schedule an appointment with a physician. The system re-encrypts Jim's record.

After finishing her consultations, Kate uploads her records to the database. The system generates a call list for Kate of those patients who need to schedule a follow-up appointment.

Patient System - Features Tested

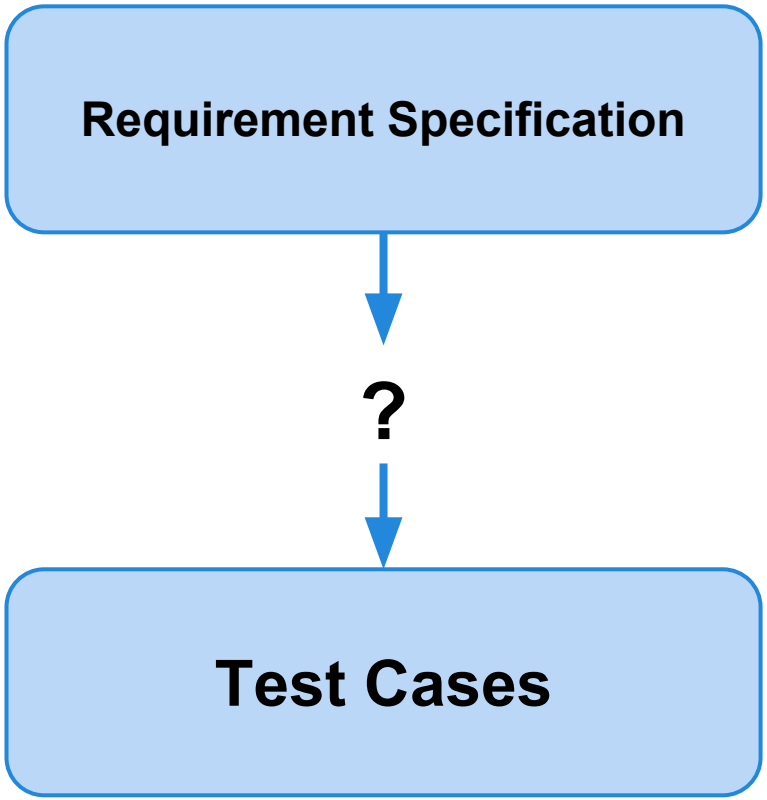
This single scenario would test:

- Authentication
- Downloading to a mobile device and uploading changes
- Home visit scheduling
- Encryption and decryption of patient records on a mobile device
- Record retrieval and modification
- Links with drug database
- System for call prompting

Outcomes of Scenario Testing

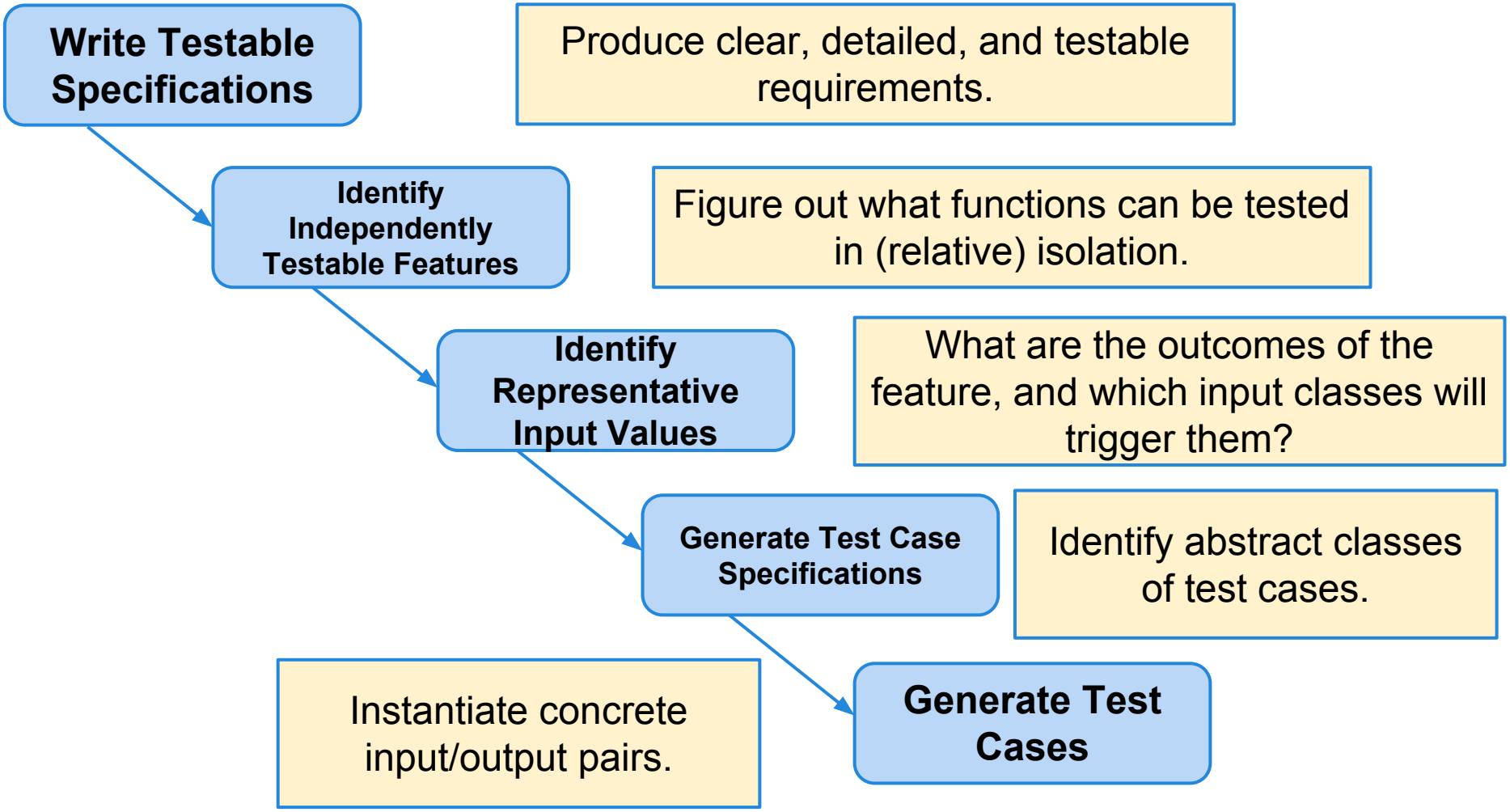
- Tester can take scenario and vary the inputs to test different outcomes.
- Each scenario covers multiple requirements, and also ensures that combinations of requirements work correctly.
- Warning -
  - Traceability is difficult. Need to maintain careful links from scenarios to requirements.
  - Need to ensure that all outcomes of software features are tested.

A Model of Testing



- Where we're at:
  - "Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system."
  - Generic scenarios that can be used as the basis for test cases.
- We need concrete test cases that can be run.
  - You can't actually test individual requirements in isolation. Need to express tests in terms of features.
  - Not all inputs have the same effect.

Creating Requirements-Based Tests



Independently Testable Feature

- Requirements are typically difficult to test in isolation. However, the system can usually be decomposed into the functions it provides.
- **An independently testable feature is a well-defined function that can be tested in (relative) isolation.**
- Identified to “divide and conquer” the complexity of functionality.

Features and Parameters

Tests for features must be described in terms of all of the parameters and environmental factors that influence the feature's execution.

- User registration on a website might take in:
  - `(firstName, lastName, dateOfBirth, eMail)`
- Consider implicit environmental factors.
  - This feature also requires a user database.
  - The existence and contents of that database influence execution.

Parameter Characteristics

The key to identifying tests is in understanding *how* the parameters are used by the feature.

- Type information is helpful.
  - `firstName` is a string, the database contains `UserRecord` structs.
- … but context is important.
  - If the database already contains an entry for that combination of fields, registration should be rejected.
  - `dateOfBirth` is a collection of three integers, but those integers are not used for any arithmetic operations.

Examples

Class Registration

What are some independently testable features?

- Add class
- Drop class
- Modify grading scale
- Change number of credits
- Graphical interface of registration page

Examples

Adding a class

What are the parameters?

- Course number to add
- Grading basis
- Student record
- What about a course database? Student record database?

Examples

GRADS

What are some independently testable features?

- Generate progress report
- Add note to a student
- Generate report with hypothetical courses added.
- Generate list of students

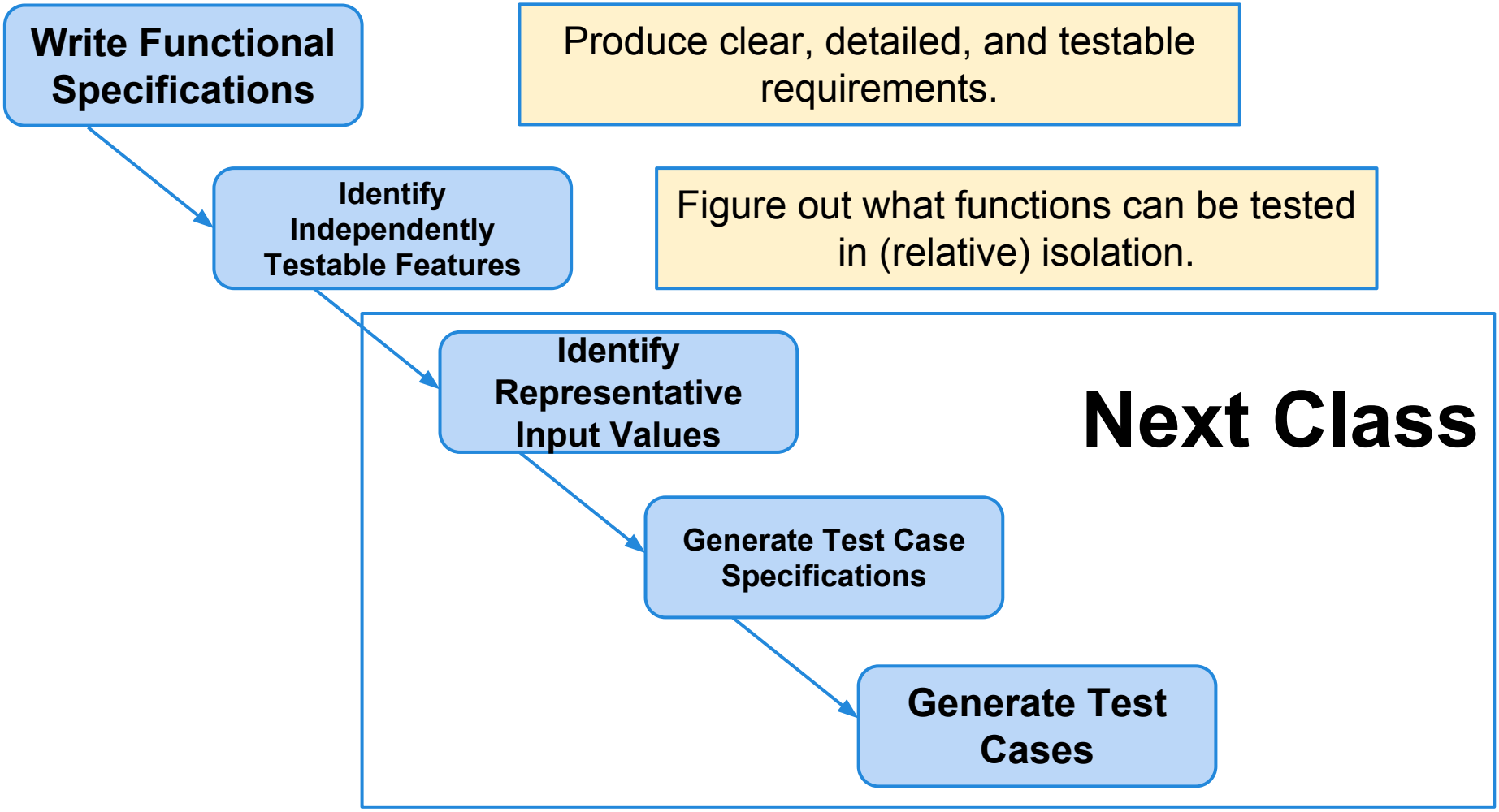
Examples

Generate progress report

What are the parameters?

- Student ID
- Record for that student
  - Records Database
- Anything else?
  - Profile of logged in user (can they access the progress report they are trying to produce?)

Where We Are At...



Key Points

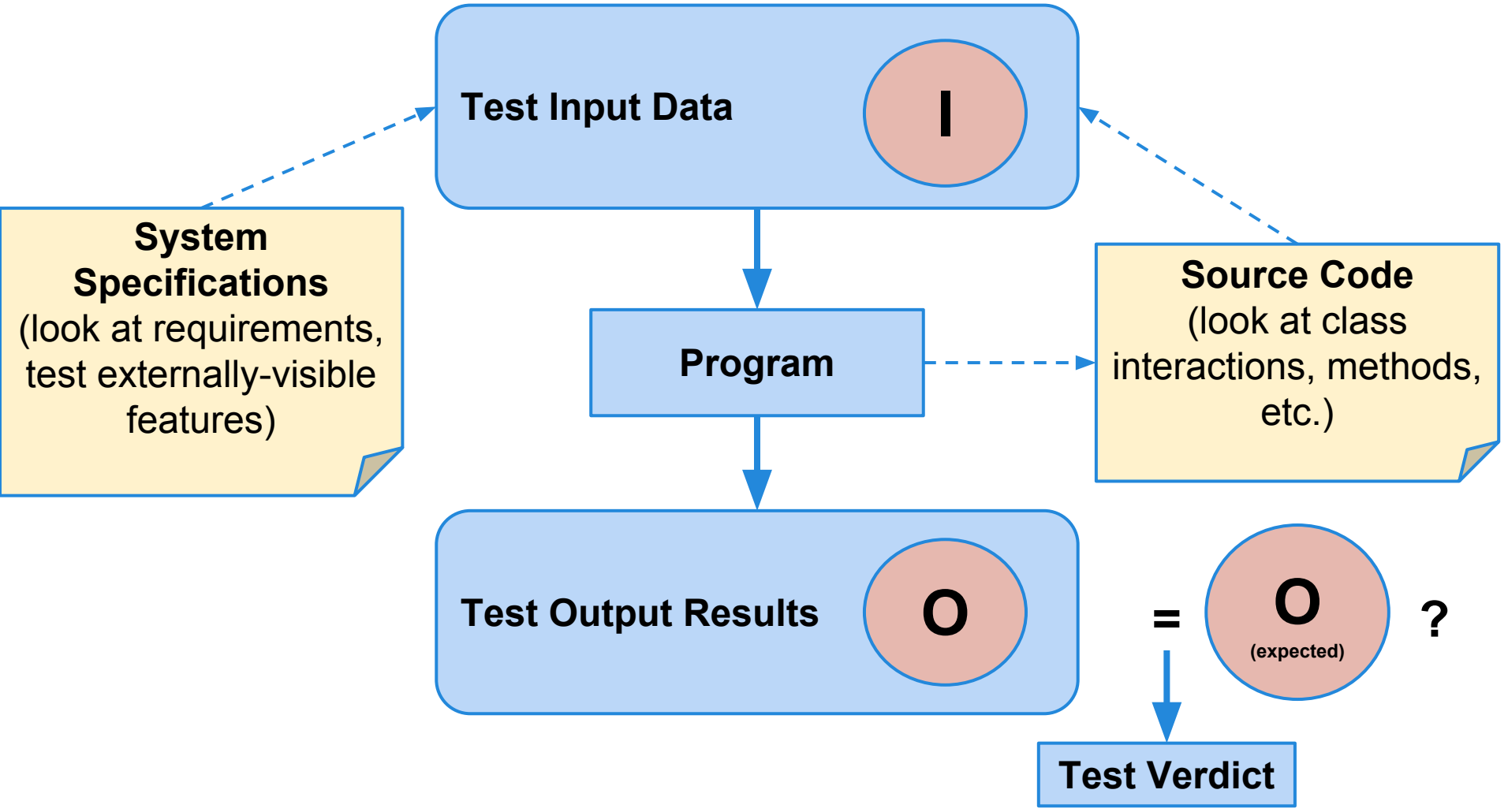
- Do yourself and the testing group a favor: **develop test cases for each requirement.**
- If the requirement cannot be tested, you most likely have a bad requirement.
  - Rewrite it so it is testable.
  - Remove the requirement if it can't be rewritten.
  - Point out why it is an unstable requirement.
- Your requirements and testing effort will be greatly improved!

Next Time

- Coming up with concrete requirements-based test cases.
- Reading:
  - Sommerville, chapter 8
    - Introduction, section 8.3.1, 8.3.2

- Homework: Draft requirements due next Wednesday!

A Model of Testing



Tailoring Tests to Requirements

More detailed, so tests should also be more objective. Can define absolute scales, exact inspections, etc.

Requ

- The
- The
- The
  into
- ...

Not written for engineers, so requirements not as detailed. Tests will be more subjective.

User Study: Can 9/10 users load the boat without help.

slide