Requirement Refinement and Testability

CSCE 247 - Lecture 7 - 02/11/2019

Today's Goals

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

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 for the requirement to be **testable**.

Why Should Requirements be Testable?

- The software might have bugs.
- *The requirements might have "bugs"*.
  - Can't "test" the document, but writing a test for the code requires thinking through the specification.
- Tests give a way to argue that the software does what we promised it would do (**verification**).
  - If a requirement is not testable, we cannot prove that the software fulfills it.

Requirements-Based Testing

- Process of deriving tests from the requirement specifications.
  - Typically the baseline technique for designing test cases. Can begin as part of requirements specification, and continue through each level of design and implementation.
  - Basis of verification - builds evidence that the implementation conforms to its specification.
  - Effective at finding some classes of faults that elude code-based techniques.
    - i.e., incorrect outcomes and missing functionality

Why Should Requirements be Testable?

- Requirements are the primary source of information to judge program behavior.
- Writing tests early:
  - Refines requirements by making them more testable.
  - Results in fewer faults when the code is written.
- Requirements-based tests can be used as evidence of verification.

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 do we make these requirements testable?

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

1. The user shall be suspended after a number of invalid attempts to enter the PIN.

Specification:

- This count shall be 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”)

Tailoring Tests to Requirement Detail Level

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

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.

Requ

- The
- The
- The
  slid
- ...

Activity: Patient Management System

Consider related requirements for a patient management system:

- If a patient is known to be allergic to any particular medication, 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.

Solution: 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 deriving tests is to use **scenarios** to develop test cases.

- Stories that describe one way in which a system might be used.
  - Use case descriptions, user stories, sequences of user interactions.
- Stories should be complex and credible.
  - Could happen in the real world, and involves complex use of the program.
- Should be easy to evaluate.
  - Scenarios are complex.

Scenario Example

- **For a Word Processor:**
- Anne is helping her daughter prepare her Girl Scout newsletter. The Girl Scouts do not mandate the content of a newsletter, but do impose requirements on the format, including the placement of logos. She positions the Girl Scout and “Girl Scout Cookie” logos in the required positions, then creates a two-column format for stories about the local group. Each story has a 18-point, bold Arial title and 12-point Arial text. She then fills in the contents of the stories. When she prints the newsletter, it appears identical to the “Print Preview” in the program.

Word Processor - Features Tested

This single scenario would test:

- Graphic importation
- Graphic placement
  - This scenario is from a real product, and did find a fault where a graphic could not be placed in a small zone in the upper-right corner.
- Creation of multi-column layout
- Font styling and size changes
- Conformance of Print Preview to final product

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

Why Use Scenario Tests?

- Learn the product.
  - People learn by investigating by themselves.
- Connect testing to requirements.
  - Complex tests are built by designing a test that runs through a series of use cases.
- Expose failures to deliver desired benefits.
  - Scenarios are end-to-end checks on a benefit a program is supposed to deliver.
  - Tests of individual features are not designed to provide this kind of check.

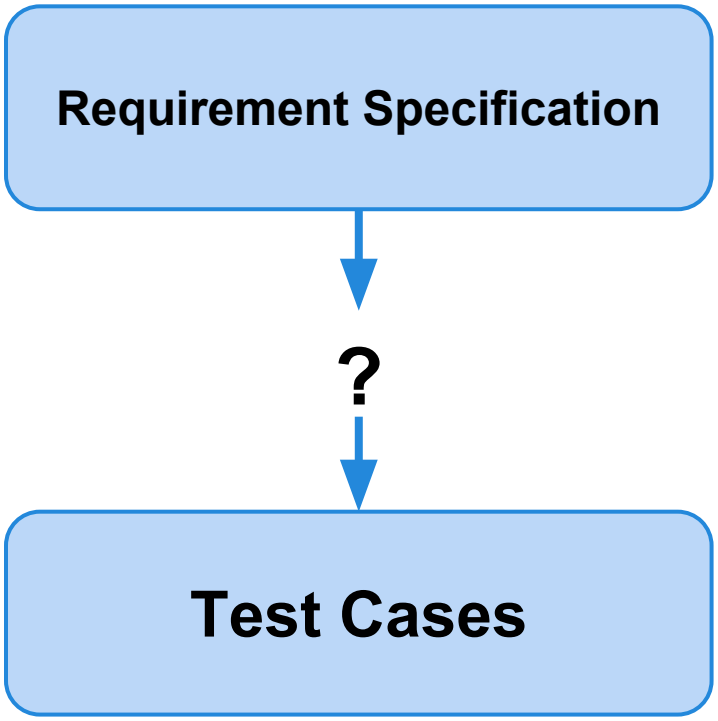
Why Use Scenario Tests?

- Explore expert use of the program.
  - People use programs differently over time.
  - Scenarios can change over time, reflecting testers growth in experience.
- Bring requirements-related issues to the surface, which might involve reopening old requirements discussions (with new data) or surfacing not-yet-identified requirements.
  - Scenario tests can highlight requirement issues or areas that were glossed over.
  - Scenarios are an early warning system for requirement problems.

Outcomes of Scenario Testing

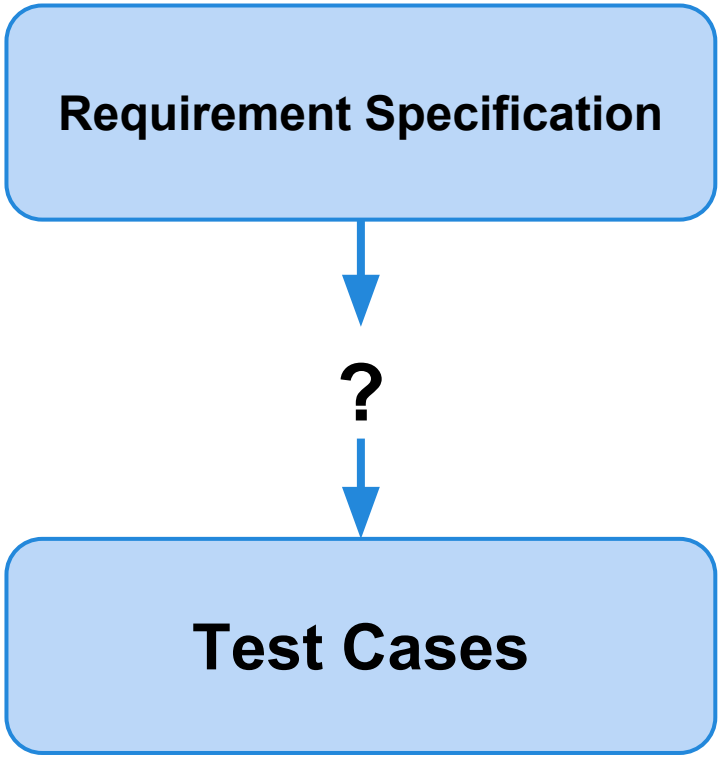
- Tester can take scenario and vary the inputs to test different outcomes.
- Each scenario covers multiple requirements, and also ensures that combinations of requirements work correctly.
- Warning -
  - Traceability is difficult. Need to maintain careful links from scenarios to requirements.
  - Need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

A Model of Testing



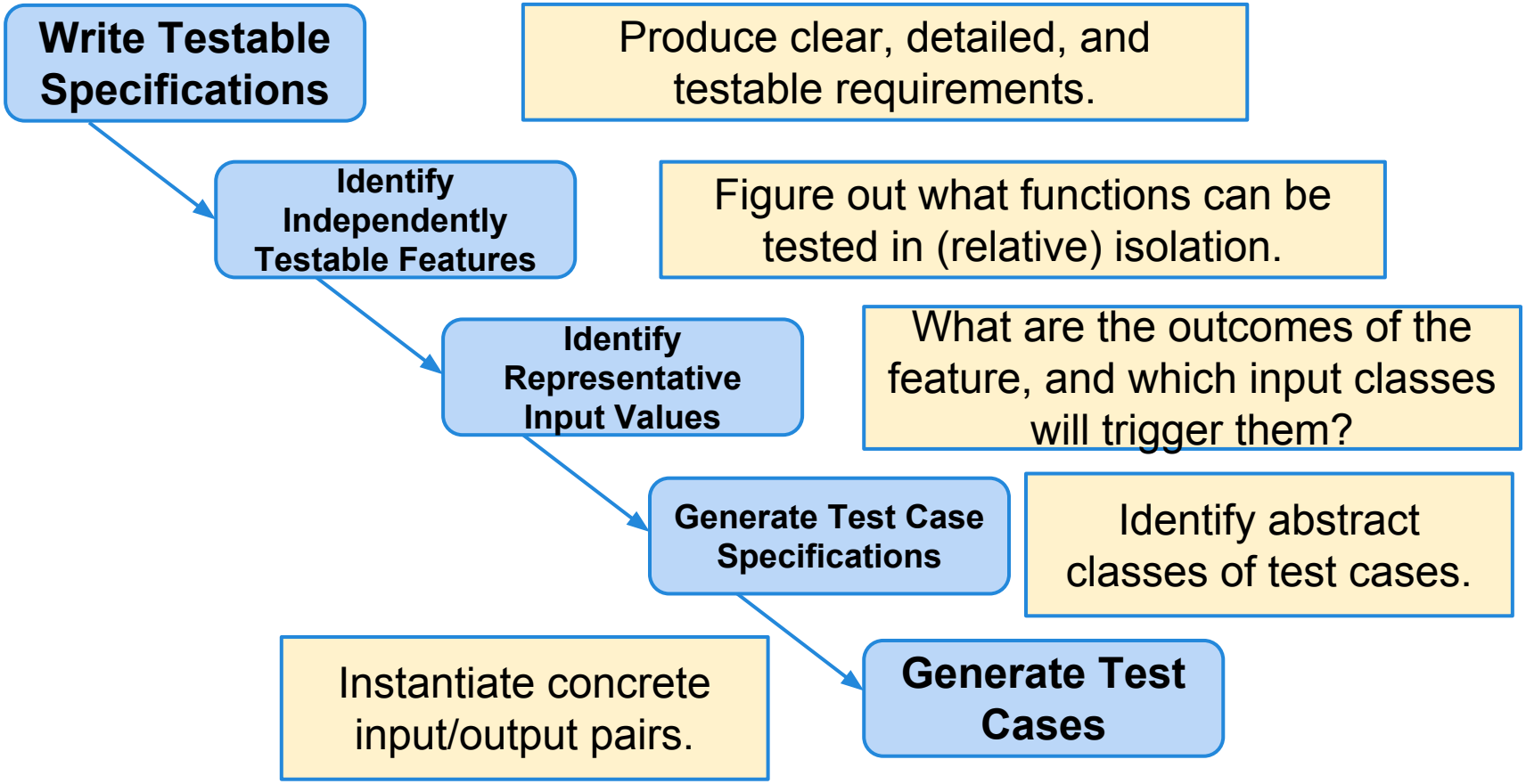
- Where we're at:
  - "Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system."
  - Generic scenarios that can be used as the basis for test cases.
- We need concrete test cases that can be run.

Partitioning



- Functional testing is based on the idea of **partitioning**.
  - You can't actually test individual requirements in isolation.
  - First, we need to partition the specification and software into features that can be tested.
  - Not all inputs have the same effect.
  - We can partition the outputs of a feature into the possible outcomes.
    - and the inputs, by what outcomes they cause (or other potential groupings).

Creating Requirements-Based Tests



Independently Testable Feature

- Requirements are 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.

Units and Features

- Executable tests are typically written in terms of “units” of code.
  - Usually a class or method.
  - Until we have a design, we do not have units.
- An independently testable feature is a *capability* of the software.
  - May not correspond to unit(s).
  - Can be at the class, subsystem, or system level.

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.

- What are the inputs to that feature?
  - User registration on a website might take in:
    - `(firstName, lastName, dateOfBirth, eMail)`
- Consider implicit environmental factors.
  - Registration 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.

Parameter Context

- An input for a feature might be split into multiple “variables” based on contextual use.
  - The database may or may not contain a record for that user.
    - In either case, issues may emerge based on the size of the database.
    - The program may also have issues if a database connection cannot be established.
  - This is three “parameters” for a feature.

Examples

Class Registration System

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

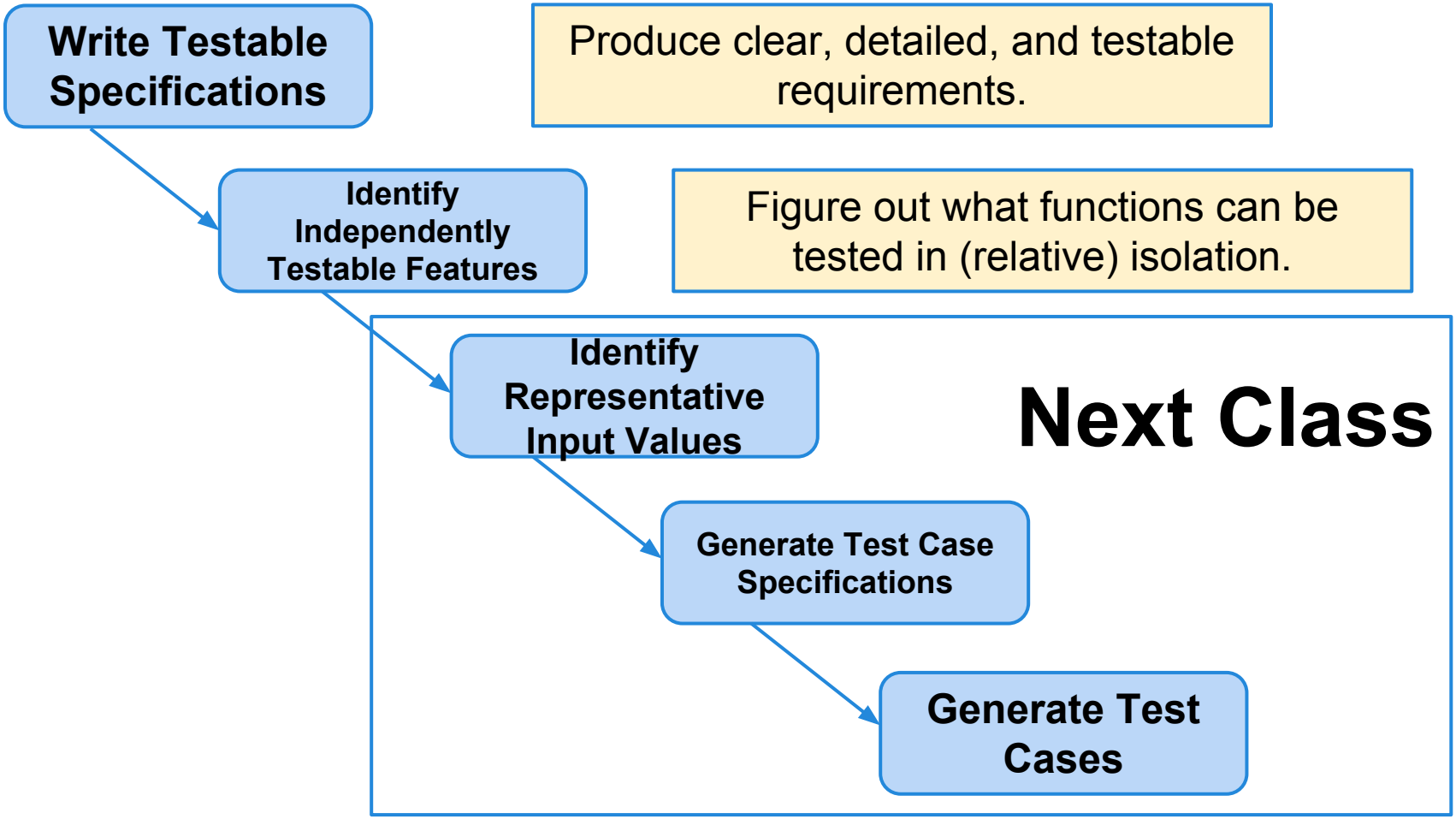
What are the parameters?

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

Examples

- Student Record
- Context - how is it used?
  - Have you already taken the course?
  - Are there holds on your record?
  - Do you meet the prerequisites?
  - …
  - Each of these can be varied when testing.

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:
  - Feedback will be out soon!
  - Assignment 2 is up.
    - Revise requirements
    - Write test cases