CS 4350: Fundamentals of Software Engineering
CS 5500: Foundations of Software Engineering

Lesson 5.1 Testing Introduction

Jon Bell, John Boyland, Mitch Wand

Khoury College of Computer Sciences

Learning Objectives for this Lesson

- By the end of this lesson, you should be able to:
  - Describe the elements of a test and how they are used;
  - State Dijkstra's law and its relevance;
  - Classify tests by purpose, scope and size;
  - Explain why test automation is important.

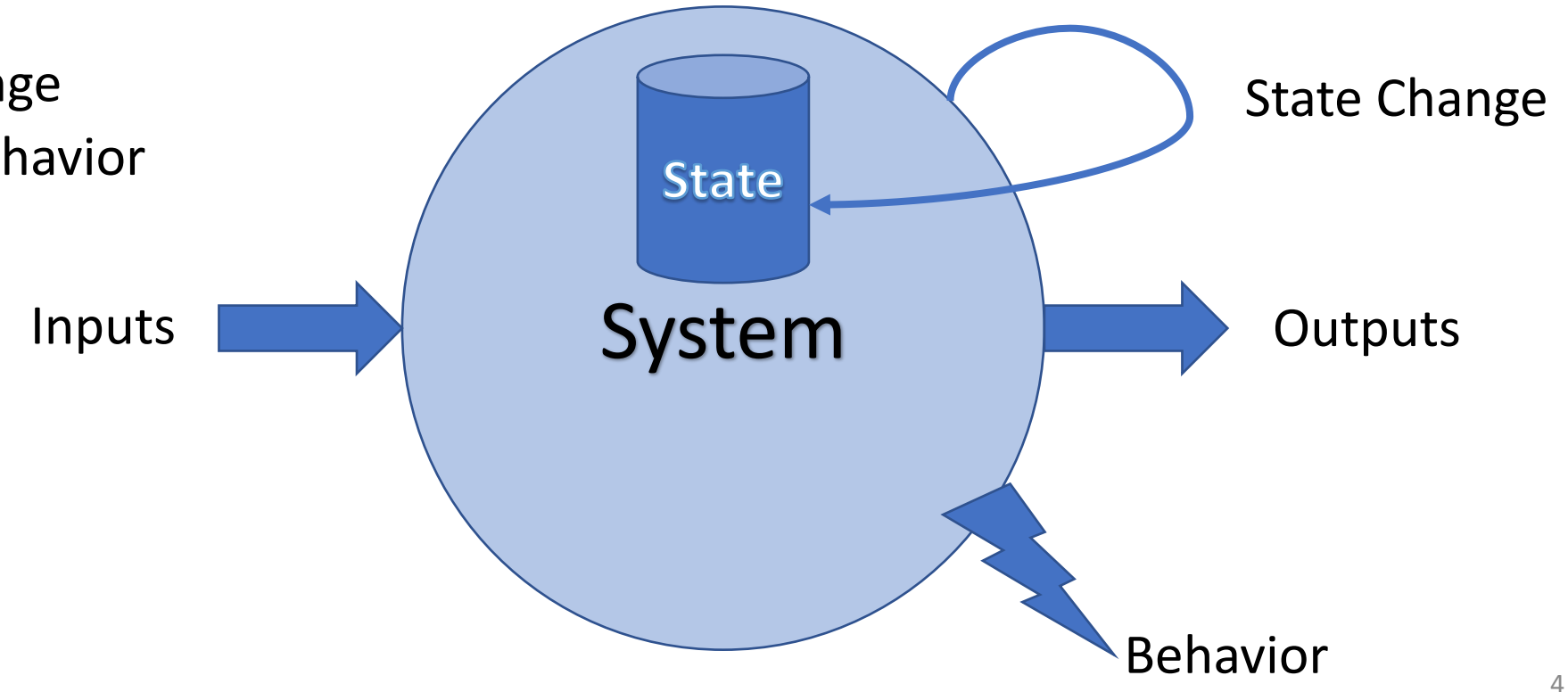
We will revisit these at the end of the lesson.

Working Definition

- *Software Testing* is the process of checking if software meets **certain concrete** requirements
  - “certain” – a finite set
  - “concrete” – particular, not symbolic
- Testing is carried out by execution of the software.
- Next: definitions “SUT” and “Test”

SUT = System Under Test

- The “System Under Test” consists of its
  - Inputs
  - State
  - Outputs
  - State Change
  - (Other) Behavior



Test

- A Test for a SUT consists of
  - Given [a certain state in the SUT],
  - When [certain inputs are presented],
  - Then [certain outputs, state change and behavior are expected].
- Example:

Given
When
Then

```
it('should set the coveyRoomID property', () => {
    const roomsStore = CoveyRoomsStore.getInstance();
    const roomName = nanoid();
    const roomController = roomsStore.getControllerForRoom(roomName);
    expect(roomController.coveyRoomID)
        .toBe(roomName);
});
```

Running a Test

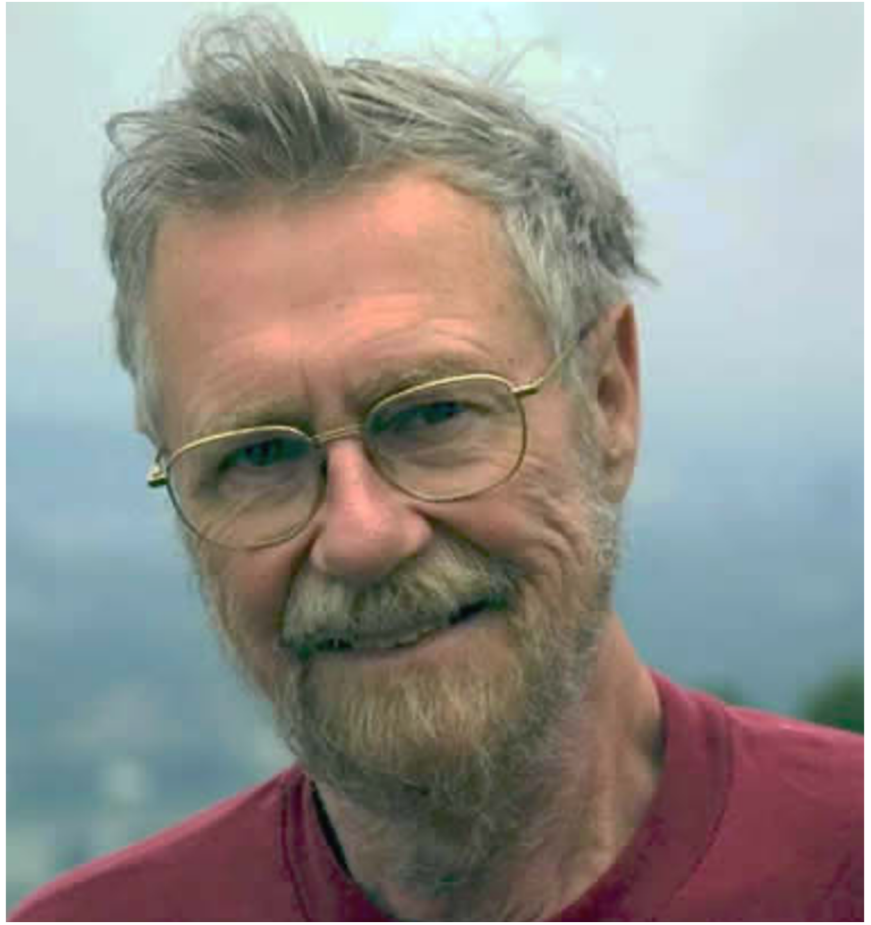
- Construct the situation:
  - Set up SUT to get the state ready
  - [Optional: Prepare collaborators]
- Apply the operation inputs.
- Check the outputs, verify the state change, handle the behavior
  - Handle exceptions,
  - Time-Out to handle nontermination,
  - Post-check with collaborators.

Dijkstra's Law

> "Program testing can be used to show the presence of bugs, but never to show their absence!"
> – Edsger Dijkstra

- The state space of a SUT is (usually) infinite, but testing can only execute a finite number of tests.
- Even if the state space is finite, it may still be too large to make exhaustive testing feasible.

And this ignores the fallibility of tests. What if the tests are in error?



Classifying Tests

- We can classify tests according to several cross-cutting dimensions:
  - Scope: What sort of thing is the SUT?
  - Purpose: Why are we testing?
  - Size: What resources does testing need?
  - How: How is testing performed?

Test Scope

- *Unit* tests: SUT = a single method/class/object
- *Integration* tests: SUT = combinations of units, a subsystem
- *System* tests: SUT = whole system being developed

Test Purpose

- Acceptance Test
  - Customer-level requirement testing
  - Validation: Are we building the right system ?
- Functional Test
  - “Black-Box” testing
  - Specification Testing
- Structural Test
  - “White-Box” testing
  - Exercising the code
- Regression Test
  - Prevent bugs from (re-)entering during maintenance.

These purposes affect how we evaluate a test suite.

Test Size

- Small: run on a single process, no blocking I/O
  - Fast to run; can be run automatically and frequently
- Medium: run on a single machine, no network I/O (only localhost); “hermetic”
  - May be slower; delayed to overnight runs
- Large/Enormous tests: run on a network.
  - May have serious $$$ cost in network services or personnel.

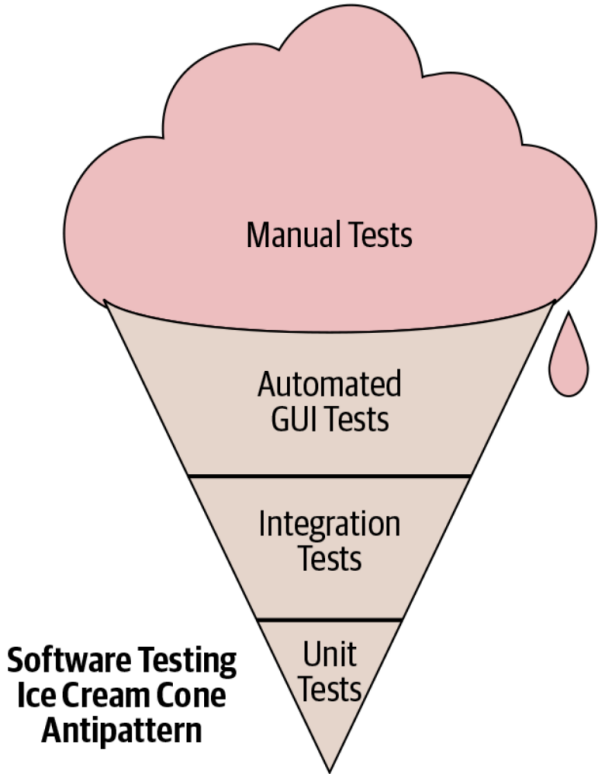
See SoftEng @ Google Chapter 11

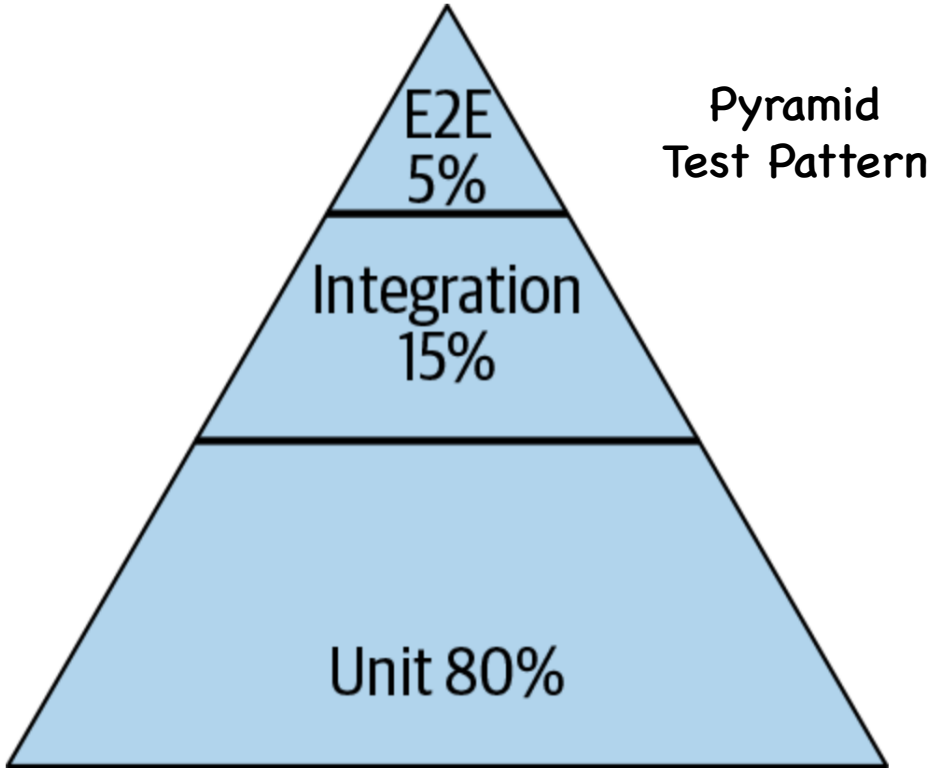
- https://learning.oreilly.com/library/view/software-engineering-at/9781492082781/ch11.html#testing_overview

Manner of Testing

- Automated tests can be run without supervision
  - Suitable for frequent automated runs
- Manual tests require a human to run and evaluate
  - A human may be needed to check UI elements
  - Tests may be ill-defined and nondeterministic
    - E.g. trying to “break” software
- Customer-facing tests require an intermediary to evaluate as well as the customer to use the software.

Test Distribution (Size/Scope/Manner)





From SoftEng @ Google Chapter 11

- https://learning.oreilly.com/library/view/software-engineering-at/9781492082781/ch11.html#testing_overview

Review

- Now that you've studied this lesson, you should be able to:
  - Describe the elements of a test and how they are used;
  - State Dijkstra’s law and its relevance;
  - Classify tests by purpose, scope and size;
  - Explain why test automation is important.

Looking Forward

- In our next lesson, we'll discuss "Test-Driven Development"