



# CS1632, Lecture 6: Test Plans

BILL LABOON

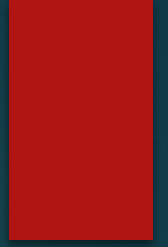
You've got requirements.  
You're looking for defects.

How?



Develop a test plan!

# Formality



- ▶ This could be as formal or informal as necessary.
- ▶ Think about what you are testing – what level of responsibility / tracking is necessary?





- ▶ Throw-away script?
- ▶ Development tool?
- ▶ Internal website?
- ▶ Enterprise software?
- ▶ Commercial software?
- ▶ Operating system?
- ▶ Avionics software?

# Testing is context-dependent

- ▶ How you test
- ▶ How much you test
- ▶ What tools you use
- ▶ What documentation you provide
- ▶ ...All vary based on software context.

# Formal Test Plans

- ▶ A test plan is a sequence of test cases.
- ▶ A test case is the fundamental “unit” of a test plan.

# A test case mainly consists of...

- ▶ Preconditions
- ▶ Execution Steps
- ▶ Postconditions

See IEEE 829, "Standard for Software Test Documentation", for more details



# Example

Assuming an empty shopping cart, when I click "Buy Widget", the number of widgets in the shopping cart becomes one.

Preconditions: User is on main page of site, with an empty shopping cart

Execution Steps: Click "Buy Widget"

Postconditions: Shopping cart displays one widget

# Example

Assuming that the SORT\_ASCENDING flag is set, calling the sort method with [9,3,4,2] will return a new array with the original data sorted from low to high, i.e., [2,3,4,9].

**Precondition:** SORT\_ASCENDING flag is set

**Execution steps:** Call .sort method with argument [9,3,4,2]

**Postconditions:** [2,3,4,9] is returned

# We also want to add:

- ▶ Identifier: A way to identify the test case
  - ▶ Could be a number
  - ▶ Often a label, e.g. INVALID-PASSWORD-THREE-TIMES-TEST
- ▶ Description: A description of the test case, describing what it is supposed to test.

# Test Plan

- ▶ These do not always test an entire system
- ▶ They may test a subsystem or related piece of functionality
  - ▶ Examples:
    - ▶ Database Connectivity Test Plan
    - ▶ Pop-up Warning Test Plan
    - ▶ Pressure Safety Lock Test Plan
    - ▶ Regression Test Plan



# Pressure Safety Lock Test Plan

LOW-PRESSURE-TEST  
HIGH-PRESSURE-TEST  
SAFETY-LIGHT-TEST  
SAFETY-LIGHT-OFF-TEST  
RESET-SWITCH-TEST  
RESET-SWITCH2-TEST  
FAST-MOVEMENT-TEST  
RAPID-CHANGE-TEST  
GRADUAL-CHANGE-TEST  
MEDIAN-PRESSURE-TEST  
LIGHT-FAILURE-TEST  
SENSOR-FAILURE-TEST  
SENSOR-INVALID-TEST

# A group of test plans make up a test suite...

- ▶ Regression Test Suite
  - ▶ Pressure Safety Regression Test Plan
  - ▶ Power Regulation Regression Test Plan
  - ▶ Water Flow Regression Test Plan
  - ▶ Control Flow Test Plan
  - ▶ Security Regression Test Plan
  - ▶ Secondary Safety Process Test Plan

# Test Run – An actual execution of a test plan or test suite.

- ▶ Analogy time: class vs object, test plan vs test run
  - ▶ The test plan is the structure, but you need to actually execute it to find out anything
- ▶ During the test run, the tester manually executes each test case and sets the status

# Possible Statuses

- ▶ PASSED
- ▶ FAILED
- ▶ PAUSED
- ▶ RUNNING
- ▶ BLOCKED
- ▶ ERROR



# Defects

- ▶ If the test case fails, a defect should be filed
  - ▶ Unless the test case has already failed, of course.
  - ▶ You don't need to re-file a duplicate of the defect!
- ▶ Note the level of formality involved will vary based on the domain

# Creating a test plan...

- ▶ Start top-down: what is a good way to subdivide the system into features (test plans)?
- ▶ For a given feature (test plan), what aspects do I want to test?
- ▶ For each aspect, what test cases do I want that will hit different equivalence classes / success or failure cases / edge or corner cases / etc.?
- ▶ How deep should I go down?
- ▶ Try to have test cases be independent of each other, and reproducible!