Software Testing Methodology Lecture 6 - BBST & BDD

Gregory S. DeLozier, Ph.D.

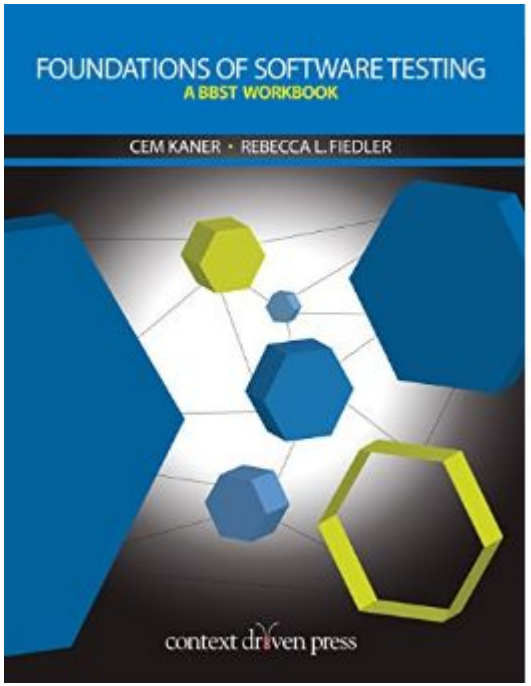
gdelozie@kent.edu

Topics

- BBST Concepts

- Behavioral Driven Design (BDD)

BBST



- "Black Box Software Testing"

- Cem Kaner
- Rebecca Fiedler

- Comprehensive discussion of testing

Software Testing...

- Is an empirical
- technical
- investigation
- conducted to provide stakeholders
- with *information*
- about the *quality*
- of the product or service under test

(Cem Kaner)

Information

- Evidence
- Cause to believe
- Probabilistic

Quality

- Value or Usefulness
- To _Someone_

- This is very subjective

Definitions

- Are not absolute
- _Are meaningful_
- You need to know what is meant _here_.

Black Box

- Can't see the internals
- Tested against external expectations
- Tested by people who know the domain
- Verify that the _system_ is correct
- (aka, "behavioral")

Glass Box

- (Not really "white box")
- Tested against internal expectations
- Does what the programmer expects
- Easier to do
- Less valuable to the consumer
- (aka, "structural)

Testing Levels

- Unit
- Integration
- System
- _Orthogonal_ to BB/GB testing
- For instance, we can unit test sorted([])

Unit

- Testing of parts
- Can happen at many levels
- Does the part work?

Integration

- Testing of many parts
- Can happen at many levels
- Can happen at levels of complexity
    - 1-1
    - Star
    - All parts

- Do the parts work together?

System

- Testing of the system
- In its environment
- Does the system meet the needs?

Functional

- Given input, proper output?

Non-functional

- Is the system behaving well while producing output?
  - usable
  - stable
  - performant
  - secure
  - etc

Acceptance testing

- Is someone going to pay for it?
- Is there a contract?
- Implied contracts
    - Development vs Marketing
    - R&D vs Product
    - Etc

Why Test?

- To gather evidence
- There are objectives
- Hard to know how well it has to work
- Hard to know how much it has to do

- Every test is a question

Oracles

- About oracles
    - Accepted definitions of correctness
    - Mathematics
    - Experts
    - Laws
- Oracles are _incomplete_
- Oracles are _heuristic_
- More on this later

Evidence?

- Evidence about quality
- Need definition of quality
    - Speed?
    - Reliability?
    - Completeness?
    - Compliance?
    - Correctness?
    - Robustness?
    - Connectivity?
- What is desirable?
- How do we measure that?

Contexts?

- Prototyping
- Mass-market Development
- Critical Environments
- Lawsuits

Testing Mission

- What are we trying to do?
- Success criteria
- Time to complete?
- Resources available?

Testing Strategy

- Given limitations
  - Time
  - Resources

- What will you do?
- How to maximize benefit?

Ex: Scenario Testing

- Complex story
- Make sure the story works

- Viewed as important

Ex: Domain Testing

- Consider all possibilities
- Group into partitions
- Select from partitions
- Select from boundaries

- Unusual tests are viewed as unlikely

Partitions

If a.x == 1:

    do_thing_a()

else:

  do_thing_b()

Then (all a where a.x == 1) is a partition

(a group treated similarly for all members)

Examples?

Techniques (James Bach's list)

Specify some or all:

- Analyze the situation
- Model the test space
- Select what to cover
- Determine test oracles
- Configure the test system
- Operate the test system
- Observe the test system
- Evaluate the results

TDD as a Technique

- Analyze the situation
    - _Feature is desired_
- Model the test space
    - _Feature gets operated_
- Select what to cover
    - _Feature examples_
- Determine test oracles
    - _Code assertions_

TDD as a Technique

- Configure the test system
  - _Unit tests_
- Operate the test system
  - _Run the tests_
- Observe the test system
  - _Assertion failures_
- Evaluate the results
  - _Code to fix the failures_

BREAK TIME

Behavior Driven Development

- A continued refinement of TDD
- Emphasize collaboration with stakeholders

- Unit tests are about specific features

- BDD tests start out with requirements
  - Emphasizing business value
  - Stated in terms of user experience

A BDD Requirement

- Example

In order to keep track of stock
As a store owner
I want to add items back to stock when they're returned.

- Includes
  - Purpose or benefit
  - Who wants it
  - What needs to happen

A BDD Criteria or Scenario

- Example

Scenario 1: Refunded items should be returned to stock
Given that a customer previously bought a black sweater from me
And I have three black sweaters in stock.
When he returns the black sweater for a refund
Then I should have four black sweaters in stock.

- Includes
  - Initial condition
  - Event or action
  - Outcome

Python 'behave' Scenario

```
Feature: showing off behave

  Scenario: run a simple test
     Given we have behave installed
      when we implement a test
      then behave will test it for us!
```

```
...this is in a _feature_ file.
```

Python 'behave' Implementation

```
from behave import *

@given('we have behave installed')
def step_impl(context):
    pass

@when('we implement a test')
def step_impl(context):
    assert True is not False

@then('behave will test it for us!')
def step_impl(context):
    assert context.failed is False
```

'behave' module

- $ pip install behave
- create file structure
  - feature file
  - ./steps/xxx.py implementation files
  - example:

```
features/
features/everything.feature
features/steps/
features/steps/steps.py
```

- $ behave

'behave' Results

```
% behave
Feature: showing off behave # tutorial/tutorial.feature:1

  Scenario: run a simple test        # tutorial/tutorial.feature:3
    Given we have behave installed   # tutorial/steps/tutorial.py:3
    When we implement a test         # tutorial/steps/tutorial.py:7
    Then behave will test it for us! # tutorial/steps/tutorial.py:11

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
```

The 'behave' Tutorial

- http://pythonhosted.org/behave/tutorial.html
- Read this
- Work through examples
- Try some examples of your own

*** THIS IS A HUGELY VALUABLE TECHNOLOGY

- There are others: cucumber, jbehave, rspec, etc.
- The ideas are very similar

DEMO TIME