Effectively Communicating with Acceptance Tests

Better Software Through Collaboration
Seminar Version
August 2015

Ken Pugh

- Fellow Consultant
- OOA&D, Design Patterns, Lean, Scrum, Test-Driven Development
- Over 2/3 century of software development experience
- Author of seven books, including:
  - Prefactoring: Extreme Abstraction, Extreme Separation, Extreme Readability (2006 Jolt Award)
  - Interface Oriented Design
  - Lean Agile Acceptance Test-Driven Development: Better Software Through Collaboration

ken.pugh
@netobjectives.com

Photo
Size:
Height: 2.25
Position: from top left corner
Horizontal 0.75
Vertical 1.
Picture Style: Simple Black Frame

Overall Rule

- There are exceptions to every statement, except this one

Flow

- Introduction
- Acceptance Test Examples
- Test Anatomy
- Supplemental
Effectively Communicating with Acceptance Tests

Introduction

5 W's and an H

- ATDD
  - What are they
  - Why use them
  - Who creates and uses them
  - When are they created
  - Where are they used
  - How to create them

What Are Acceptance Tests?

- Acceptance Tests:
  - External view of system
- Examine externally visible effects
  - Inputs and outputs
  - State changes
  - External interfaces

Definitions

- Acceptance criteria
  - General ideas
- Acceptance tests
  - Specific tests that either pass or fail
  - Implementation independent
- Triad – customer unit, developer unit, tester unit
Effectively Communicating with Acceptance Tests

Fast Car Example

- Who wants a fast car?
- Criteria
  - Run on a closed course, measure acceleration
- Test
  - Detail acceleration (0 to 60 mph in X seconds)

ATDD as a Prism

Requirements and Tests

- Requirements and tests are inter-related
  - You can’t have one without the other
- A failing test is a requirement
  - Passing test denotes specification on how system works

Why?

- Rework Down from 60% to 20%
- Workflows Working First Time
- Little Room for Miscommunication
- Saving Time
- Getting Business Rules Right
- Game Changing
- Tighter Cross-Functional Team Integration
- Crisp Visible Story Completion Criteria
- Automation Yields Reduced Testing Time
Effectively Communicating with Acceptance Tests

Value Stream Map – Classical Development

- Elicit Requirements → Analyze Requirements → Design
- Code → Test → Deploy

Why go back?

Value Stream Map – Agile Development

- Elicit Requirements → Analyze Requirements With Tests → Design
- Code With Tests → Deploy

Why Mistakes?

- Misunderstandings, missed requirements, misplaced
- Feedback helps to correct misunderstandings
- Quick feedback better than slow feedback

Move Testing Forward

- Two types of testing
  - Attempting to find defects – is WASTE
  - Attempting to prevent defects – is ESSENTIAL
- When are defects found?
  - Prevention is just early detection
Effectively Communicating with Acceptance Tests

**Cost of a Requirement Issue**

- **Effort**: Could be 1 to 64, 1 to 256, or something else

**Who Does What**

- **Author the tests (write)**
  - Triad - customer, tester, developer
- **Connect tests to system (bind)**
  - Developer
- **Run the tests (run)**
  - Developers, testers, customers
  - Automated – part of build

**ATDD, TDD, BDD**

- **ATDD**
  - Tests written by triad prior to implementation
  - Implementation independent, can be automated or manual
  - Usually uses Given/When/Then template
- **TDD** – Test Driven Development/ Test Driven Design
  - Done by developer while coding
  - Implementation dependent, always automated
  - Write unit test, see it fail, implement code to pass
- **BDD** – Behavior Driven Development
  - Started as replacement for TDD unit testing framework
  - Usually associated with Cucumber-like languages (Given/When/Then)
  - Otherwise, like ATDD
- **SPE** – Specification by Example
  - Like ATDD, except uses “examples”, rather than “tests”

**Acceptance Test Examples**
First Example

- Input Temperature in Celsius, Output Temperature in Fahrenheit
- What tests would you run?

Input and Output Example (continued)

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>212</td>
<td>How many needed?</td>
</tr>
<tr>
<td>-273.15</td>
<td>-459.67</td>
<td>2 digits precision</td>
</tr>
<tr>
<td>-273.151</td>
<td>Error</td>
<td>Below 0 Kelvin</td>
</tr>
<tr>
<td>500</td>
<td>932</td>
<td>Maximum – Needed?</td>
</tr>
</tbody>
</table>

Input and Output – “Unit Tests”

Formula Tests

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>212</td>
<td>How many needed?</td>
</tr>
</tbody>
</table>

Precision Tests

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-273.15</td>
<td>-459.67</td>
<td></td>
</tr>
</tbody>
</table>

Limit Tests

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-273.15</td>
<td>-459.67</td>
<td>0 Kelvin</td>
</tr>
<tr>
<td>-273.151</td>
<td>Error</td>
<td>Below 0 Kelvin</td>
</tr>
<tr>
<td>500</td>
<td>932</td>
<td>Maximum – Needed?</td>
</tr>
</tbody>
</table>

A Business Rule Example

If Customer Rating is Good and the Order Total is less than or equal $10.00,
Then do not give a discount,
Otherwise give a 1% discount.

If Customer Rating is Excellent,
Then give a discount of 1% for any order.
If the Order Total is greater than $50.00,
Then give a discount of 5%.

Given a customer whose rating is Good and an order total of $50.01, what should be the discount?
Effectively Communicating with Acceptance Tests

Business Rule Table = Test

<table>
<thead>
<tr>
<th>Discount</th>
<th>Order total</th>
<th>Customer rating</th>
<th>Discount percentage?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10.00</td>
<td>Good</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>$10.01</td>
<td>Good</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>$50.01</td>
<td>Good</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>$.01</td>
<td>Excellent</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>$50.00</td>
<td>Excellent</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>$50.01</td>
<td>Excellent</td>
<td>5%</td>
<td></td>
</tr>
</tbody>
</table>

Ways To Implement Test

- Testing script
- Program interface
- Xunit framework
- ATDD framework

Testing script

- Tester creates script (usually GUI based)
- Example:
  - Logon as Customer who is rated Good
  - Start order
  - Put items in the order until the total is exactly $10.01
  - Complete order
  - Check it shows a $.10 discount
- Repeat for other five cases

Program interface

- Create a command line or graphic user interface

```
C:\>DiscountPercentage Good 10.01
Discount Method
Screen
Discount Percentage Screen
Customer Type: Good
Order Total: 10.01
Percentage: 1 %
```
Effectively Communicating with Acceptance Tests

XUnit Test

class TestCase {
    testDiscountPercentageForCustomer() {
        SomeClass o = new SomeClass()
        assertEquals(0, o.computeDiscount(10.0, Good));
        assertEquals(1, o.computeDiscount(10.01, Good));
        assertEquals(1, o.computeDiscount(50.01, Good));
        assertEquals(1, o.computeDiscount(.01, Excellent));
        assertEquals(5, o.computeDiscount(50.0, Excellent));
    }
}

Fit (Table = Test)

<table>
<thead>
<tr>
<th>Discount</th>
<th>Order total</th>
<th>Customer rating</th>
<th>Discount percentage?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10.00</td>
<td>Good</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>10.01</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50.01</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>.01</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50.00</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50.01</td>
<td>Excellent</td>
<td>5</td>
</tr>
</tbody>
</table>

Fit Test

<table>
<thead>
<tr>
<th>Discount</th>
<th>Order total</th>
<th>Customer rating</th>
<th>Discount percentage?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$10.00</td>
<td>Good</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>$10.01</td>
<td>Good</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>$50.01</td>
<td>Good</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>$.01</td>
<td>Excellent</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>$50.00</td>
<td>Excellent</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>$50.01</td>
<td>Excellent</td>
<td>5%</td>
</tr>
</tbody>
</table>

Tables As Requirement and Test

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Discount Rule</th>
<th>Order Total</th>
<th>Discount Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>&lt;= $10.00</td>
<td>Good</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Otherwise</td>
<td>Any</td>
<td>1%</td>
</tr>
<tr>
<td>Excellent</td>
<td>Any</td>
<td>$50.00</td>
<td>5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>Discount Test</th>
<th>Order total</th>
<th>Customer rating</th>
<th>Discount percentage?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$10.00</td>
<td>Good</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$10.01</td>
<td>Good</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$50.01</td>
<td>Good</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$.01</td>
<td>Excellent</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$50.00</td>
<td>Excellent</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$50.01</td>
<td>Excellent</td>
<td>5%</td>
<td></td>
</tr>
</tbody>
</table>
Effectively Communicating with Acceptance Tests

Cucumber Version

Scenario Outline: Compute discount
Given total is <OrderTotal> and rating is <CustomerRating>
When I compute discount
Then percent is <DiscountPercentage>
Examples:
|OrderTotal|CustomerRating|DiscountPercentage|
|10.00     |Good        |0                     |
|10.01     |Good        |1                     |
|50.01     |Good        |1                     |
|0.01      |Excellent   |1                     |
|50.00     |Excellent   |1                     |
|50.01     |Excellent   |5                     |

Test Anatomy

Scenario Flow

Example of Scenario

- Given (Setup)
  - Car is not moving
- When (Trigger)
  - Accelerator pressed
- Then (Assert)
  - 60 MPH reached before 4.5 seconds
Effectively Communicating with Acceptance Tests

Test Flow

- **Given (Setup)**
  - Car is not moving
- **When (Trigger)**
  - Accelerator pressed
- **Then (Verify)**
  - Check that 60 MPH is reached before 4.5 seconds

Expected system state and output = behavior
- Expected behavior drives development → Behavior-Driven Development
- Tests that behavior is acceptable → Acceptance tests drive development → Acceptance Test-Driven Development

Term Alternatives

Workflow Example
Given / When / Then Example

- Context: Sam’s Lawn Mower Repair and CD Rental Store
- Given (Setup)
  - Customer has ID (initial system state)
  - CD has ID (initial system state)
  - CD is not currently rented (initial system state)
- When (Trigger)
  - Clerk checks out CD (action)
- Then (Verify)
  - CD recorded as rented (final system state)
  - Rental contract printed (output)

Full Example (1)

- **Check Out CD**
- Given Customer has ID and CD has ID and CD is not currently rented

<table>
<thead>
<tr>
<th>Customer Data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>James</td>
<td></td>
<td>007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CD Data</th>
<th>ID</th>
<th>Title</th>
<th>Rented</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD2</td>
<td></td>
<td>Beatles Greatest Hits</td>
<td>No</td>
</tr>
</tbody>
</table>

Full Example (2)

- When a clerk checks out a CD:

<table>
<thead>
<tr>
<th>Check Out CD</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer ID</td>
<td>007</td>
<td></td>
</tr>
<tr>
<td>Enter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD ID</td>
<td>CD2</td>
<td></td>
</tr>
<tr>
<td>Execute</td>
<td>Rent</td>
<td></td>
</tr>
</tbody>
</table>

Full Example (3)

- Then the CD is recorded as rented and a rental contract is printed:

<table>
<thead>
<tr>
<th>CD Data</th>
<th>ID</th>
<th>Title</th>
<th>Rented</th>
<th>Customer ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD2</td>
<td></td>
<td>Beatles Greatest Hits</td>
<td>Yes</td>
<td>007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rental Contract</th>
<th>Customer ID</th>
<th>Customer Name</th>
<th>CD ID</th>
<th>CD Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>007</td>
<td>James</td>
<td>CD2</td>
<td></td>
<td>Beatles Greatest Hits</td>
</tr>
</tbody>
</table>
Full Example – Extended

- **Given**
  - Rental Fee Business Rule
    - Fee: $3
  - Rental Time Business Rule
    - Time: 2 days

- When a clerk checks out a CD on:
  - Today: 1/1/2014

- Then a rental contract is printed:

<table>
<thead>
<tr>
<th>Rental Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer ID</td>
</tr>
<tr>
<td>007</td>
</tr>
</tbody>
</table>

The Action

- Can drive a GUI
  - ![GUI example](image)

- Or a method
  - `Rent(CustomerID aCustomer, CDID aCD);`

- Or an Interactive Voice Response (IVR)
  - “Enter the customer id followed by the pound sign”

Example of Business Rule

**CD Rental Rates Business Rule**
- Regular $2 / 2 days plus $1 / day
- Golden Oldie $2 / 4 days plus $.50 / day
- Hot Stuff $2 / 1 days plus $2 / day

<table>
<thead>
<tr>
<th>CD Rental Rates As Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Regular</td>
</tr>
<tr>
<td>Golden Oldie</td>
</tr>
<tr>
<td>Hot Stuff</td>
</tr>
</tbody>
</table>

Example of Business Rule Test

**CD Rental Rates Business Rule**
- Regular $2 / 2 days plus $1 / day
- Golden Oldie $2 / 4 days plus $.50 / day
- Hot Stuff $2 / 1 days plus $2 / day

<table>
<thead>
<tr>
<th>CD Rates Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Regular</td>
</tr>
<tr>
<td>Golden Oldie</td>
</tr>
<tr>
<td>Hot Stuff</td>
</tr>
<tr>
<td>Hot Stuff</td>
</tr>
</tbody>
</table>
Business Rule and Flow Tests

- Use cases/scenarios usually include business rules
- Test every business rule separately
- Flow test of scenario
  - Uses one variation of business rule

Full Flow Example Revisited

**Check Out CD**
Given Customer has ID; CD has ID and not currently rented

<table>
<thead>
<tr>
<th>Customer Data</th>
<th>CD Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ID</td>
</tr>
<tr>
<td>James</td>
<td>007</td>
</tr>
</tbody>
</table>

and one variation of business rule for Rental Rates

<table>
<thead>
<tr>
<th>CD Rental Rates As Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Regular</td>
</tr>
</tbody>
</table>

Full Flow Example – Extended

- When a clerk checks out a CD on:
  - Check Out CD
    - Enter Customer ID: 007
    - Enter CD ID: CD2
    - Execute Rent

- Then a rental contract is printed:

<table>
<thead>
<tr>
<th>Rental Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer ID</td>
</tr>
<tr>
<td>007</td>
</tr>
</tbody>
</table>
Effectively Communicating with Acceptance Tests

Tests

- Acceptance tests are not a substitute for interactive communication
  - They can provide focus for that communication
- Tests in business domain terms
  - Shared between customer unit and developer unit

Guidelines

- Tests and automation should be developed separately
  - Understand the test first
  - Then explore how to automate it
- Automate tests for regression
  - Use in continuous build
- As much as practical, cover 100% of the functional requirements by acceptance tests
- Can break down stories by acceptance tests
  - One acceptance test per story

A Few Other Thoughts

Testing Strategies

![Testing Strategies Diagram](Image)
Effectively Communicating with Acceptance Tests

Context Diagram

- A context diagram shows scope

Context of Automobile

- Acceleration
- De-acceleration
- Movement

Component tests for the individual pieces

External Interfaces

- Connections to external systems need to have test doubles (mocks)
  - They are stand-ins for real system during testing
  - Random events may need to simulated
- Test doubles give repeatability and speed
  - Cheaper if have to pay for service

Tests to Cover Entire Flow

- UI
- Application / Middle tier
- Core Business Logic
- Persistence
- Service
Effectively Communicating with Acceptance Tests

Automation Testing Pyramid

Avoid Copy and Paste
In requirements, tests, code, documentation

Recap

- Primary goals
  - Discover ambiguous requirements and gaps in requirements early on
  - Create a record of business/development understanding
- Secondary goals
  - Use acceptance tests as an executable regression test
  - Measure the complexity of requirements
  - Use the tests as a basis for user documentation

Not an Ending, But a Beginning
Go Forth and Become Acceptance Test Creators

Thank You