Accelerating DevOps with Behavior-Driven Development (BDD)

Ken Pugh
Ken Pugh

ASE, BDD/ATDD, TDD, SAFe (SPC4), Lean, Scrum, Kanban, Design Patterns
Training and Consulting

http://atdd-bdd.com
ken@kenpugh.com
kenpugh
@kpugh

Lean Agile Acceptance Test-Driven Development:
Better Software Through Collaboration
There are exceptions to every statement, except this one
Second Overall Rule

Context is everything

Everything exists in a context
Everything is always true in some context
ATTD/BDD Specific Rule

No code goes in till the test goes on
Introduction to Acceptance Tests/Behavior Driven Development
What Are Acceptance Tests?

Tests from external view of system
Definitions

- Acceptance criteria
  - General ideas

- Acceptance tests
  - Specific tests that either pass or fail
  - Implementation independent

- Triad – customer, developer, tester perspectives
Fast Car Example

Who wants a fast car?

- Acceleration 0 to 60 in X seconds
- Top speed Y mph
- Time at top speed Z seconds
BDD Scenario Template

- **Given** (Setup) → Initial system state
- **When** (Trigger) → Action or event occurs
- **Then** (Assert) → New system state
  - Output
Given (Setup)
   Car is not moving

When (Trigger)
   Accelerator pressed

Then (Assert)
   60 MPH reached before X seconds
Term Alternatives

- 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
Example

Acceleration

Movement

Implementation Independent

Noise

Automobile
BDD Discovery

Fast Car

Requirement

Triad

Scenario
Acceleration

Scenario
Top Speed

Scenario
More
DevOps without BDD/ATDD
DevOps without BDD / ATDD

DEFECT: Not working right ☢️
Why Loopbacks?

Misunderstandings, missed requirements, mis-other

Feedback helps to correct misunderstandings

Quick feedback better than slow feedback
Example of BDD
Sample Business Rule

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%.

What discount for a Good Customer and $50.01 Order Total?
1% ?
5% ?
6% ?
### Example

<table>
<thead>
<tr>
<th>Given</th>
<th>Discount</th>
<th>When Then</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Order total</strong></td>
<td><strong>Customer rating</strong></td>
<td><strong>Discount percentage?</strong></td>
</tr>
<tr>
<td>$50.01</td>
<td>Good</td>
<td>1%</td>
</tr>
<tr>
<td>$10.00</td>
<td>Good</td>
<td>0%</td>
</tr>
<tr>
<td>$10.01</td>
<td>Good</td>
<td>1%</td>
</tr>
<tr>
<td>$.01</td>
<td>Excellent</td>
<td>1%</td>
</tr>
<tr>
<td>$50.00</td>
<td>Excellent</td>
<td>1%</td>
</tr>
<tr>
<td>$50.01</td>
<td>Excellent</td>
<td>5%</td>
</tr>
</tbody>
</table>
Ways To Implement Test

- Testing script
- Xunit framework
- BDD/ATDD framework
Creating the Script

Tester creates script (usually GUI based), e.g.:

1. Logon as Customer who is rated Good
2. Start order
3. Put items in the order until the total is exactly $50.01
4. Complete order
5. Check it shows a $.50 discount

Repeat for other five cases
class TestCase {

    void testDiscountPercentageForCustomer() {
        SomeClass o = new SomeClass();
        assertEquals(1, o.computeDiscount(50.01, Good));
        assertEquals(0, o.computeDiscount(10.00, Good));
        assertEquals(1, o.computeDiscount(10.01, Good));
        assertEquals(1, o.computeDiscount(00.01, Excellent));
        assertEquals(5, o.computeDiscount(50.00, Excellent));
        assertEquals(1, o.computeDiscount(50.01, Excellent));
    }
}
## Example

<table>
<thead>
<tr>
<th>Discount</th>
<th>Order total</th>
<th>Customer rating</th>
<th>Discount percentage?</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.01</td>
<td>50.01</td>
<td>Good</td>
<td>Expected 1 Actual 5</td>
</tr>
<tr>
<td>10.00</td>
<td>10.00</td>
<td>Good</td>
<td>0</td>
</tr>
<tr>
<td>10.01</td>
<td>10.01</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td>50.00</td>
<td>50.00</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td>50.01</td>
<td>50.01</td>
<td>Excellent</td>
<td>5</td>
</tr>
</tbody>
</table>
**Scenario Outline: Compute discount**

Given total is <Order Total> and rating is <Customer Rating>

When discount computed

Then percent is <Discount Percentage>

**Examples:**

<table>
<thead>
<tr>
<th>Order Total</th>
<th>Customer Rating</th>
<th>Discount Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.01</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td>10.00</td>
<td>Good</td>
<td>0</td>
</tr>
<tr>
<td>10.01</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td>0.01</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td>50.00</td>
<td>Excellent</td>
<td>1</td>
</tr>
<tr>
<td>50.01</td>
<td>Excellent</td>
<td>5</td>
</tr>
</tbody>
</table>
Testing Pyramid

Testing script (1)

BDD/ATDD (6)
<table>
<thead>
<tr>
<th>Customer Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
</tr>
<tr>
<td>Excellent</td>
</tr>
<tr>
<td>Anything else?</td>
</tr>
</tbody>
</table>
BDD Workflow Example
Example Scenario

- **Given (Setup)**
  - Customer has ID (initial system state)
  - Album has ID (initial system state)
  - Album is not currently rented (initial system state)

- **When (Trigger)**
  - Clerk checks out Album (action)

- **Then (Assert)**
  - Album recorded as rented (final system state)
  - Rental contract printed (output)
Flow Test 1

Given Customer has ID

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

and Album has ID
and Album is not currently rented

<table>
<thead>
<tr>
<th>Album Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>A2</td>
</tr>
</tbody>
</table>
When a clerk checks out an Album:

<table>
<thead>
<tr>
<th>Check Out Album</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter</td>
<td>Enter Customer ID</td>
<td>007</td>
</tr>
<tr>
<td>Enter</td>
<td>Enter Album ID</td>
<td>A2</td>
</tr>
<tr>
<td>Execute</td>
<td>Execute CheckOut</td>
<td></td>
</tr>
</tbody>
</table>
Then the Album is recorded as rented

<table>
<thead>
<tr>
<th>Album Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>A2</td>
</tr>
</tbody>
</table>

and 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>
Full Example – Extended

- **Given**
  
<table>
<thead>
<tr>
<th>Rental Fee Business Rule</th>
<th>Rental Time Business Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fee</td>
<td>Time</td>
</tr>
<tr>
<td>$3</td>
<td>2 days</td>
</tr>
</tbody>
</table>

- **When** a clerk checks out an Album on:
  
  | Today  | 6/1/2018 |

- **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

  ![Check-out](Check-out.png)

  - Customer ID
  - CD ID

  - Rent

- Or a method

  `CheckOut(CustomerID aCustomer, AlbumID aAlbum);`

- Or an Interactive Voice Response (IVR)

  "Enter the customer id followed by the pound sign"
The Action

- Values in Then come from
  - Given
  - When
  - Business Rules
Given are somebody’s then:

- **Register User**
  - Then **Customer has ID**

- **Add Album to Inventory**
  - Then **Album has ID**
  - **Album is not currently rented**

- **Checkout Album**

---

© Scaled Agile, Inc. and Ken Pugh, Inc.
Requirements and Tests
Types of Testing

- Two types of testing
  - Attempting to find defects
  - Attempting to prevent defects

- When are defects found?
  - Prevention is just early detection
Requirements and Tests

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

BDD Test

Create

Connect

Check

Triad

Developer

Develop, Build, Regression
Requirements and Test

One Way

Code → Create → Connect → Check

Better Way

Create → Connect → Check → Code
Cost of Requirement Issue

Could be 1 to 64, 1 to 256, or something else
Not an Ending, But a Beginning
DevOps with BDD/ATDD
DevOps with BDD/ATDD

Replace misunderstanding with Shared understanding
- Right Tests
  - Rework Down from 60% to 20%
  - Getting Business Rules Right
  - Zero Production Defects
  - Crisp Visible Story Completion
  - Tighter Cross-Functional Team Integration

- Automate Right
  - Accelerates DevOps
Recap

- Primary goals
  - Discover ambiguous requirements and gaps in requirements early on
  - Create a record of business/development understanding

- Secondary goals
  - Measure the complexity of requirements
  - Use the tests as basis for documentation
Recap

- Report from team 4 months after BDD/ATDD adoption:
  - Team “happiness factor” increased
    - Specifically, lead developer and tester are much happier
    - Less stress on testers
  - More distributed testing effort across the sprint
  - Helped to create/enhance “we are a team” feeling
  - Fewer production defects
  - Fewer test environment defects
  - Less rework due to miscommunication
Go Forth and Become Behavior/Acceptance Test Creators

Thank you
Thank you!
Supplemental
<table>
<thead>
<tr>
<th>Tests</th>
<th>Kind of Behavior</th>
<th>Per Functionality</th>
<th>Cross Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Facing</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology Facing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer/User Tests</td>
<td>Business Intent</td>
<td></td>
<td>Usability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Is it pleasurable?</td>
</tr>
<tr>
<td>Component Tests</td>
<td>Architectural Intent</td>
<td></td>
<td>Exploratory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Is it self-consistent?</td>
</tr>
<tr>
<td>Unit Tests</td>
<td>Developer Intent</td>
<td></td>
<td>Property Testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Is it scalable, secure, responsive?</td>
</tr>
</tbody>
</table>

Adapted from Mary Poppendieck, Brian Merrick, and Gerard Meszaros
ATDD/BDD
Ken Pugh

Temperature Example
Try It Out

- Input temperature in Celsius, output temperature in Fahrenheit
- What tests would you run?

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>???</td>
<td>Freezing</td>
</tr>
</tbody>
</table>
### 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>Precision</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></td>
<td></td>
<td>Error</td>
</tr>
<tr>
<td>-273.16</td>
<td>Error</td>
<td>Below 0 Kelvin</td>
</tr>
<tr>
<td>500</td>
<td>932</td>
<td>Maximum – Needed?</td>
</tr>
</tbody>
</table>