BDD, ATDD, TDD, DDD, DP, OOAD, ... If You’re D’d out, Try A Holistic Approach

Ken Pugh
ken@kenpugh.com

Who are you?
• Developers, testers, others?
• Why are you here?

Objectives
• Explain the different contexts of an application
• Identify what are external and internal behaviors and how to specify them
• Create a shared understanding of the behavior of business rules and domain terms
• Minimize the amount of redundancy in describing behavior

Path
• External Behavior
• Internal Behavior
• Domain Term and Business Rule Behavior
• Automation of Behavior Tests
• Quality Attributes for Behavior

Introduction

Ken Pugh
• ATDD/BDD, TDD, BVDD, Lean, Scrum, SAEFe, Design Patterns
• Over 2/5 century of software development experience
• Co-author SAFe® Agile Software Engineering
• Author of seven books

ken@kenpugh.com
https://www.linkedin.com/in/kenpugh/
http://acceptancetestdrivendevelopment.com
Overall Rule

There are exceptions to every statement, except this one

2nd Overall Rule

Context is everything
Everything exists in a context
Everything is always true in some context

Always Tradeoffs

• Commonality, e.g. libraries versus
• Dependence on those libraries

Definitions

• Component
  Service, Module, Class

• Responsibility
  Fulfilled by cohesive set of behaviors
Context of Behavior

- What is inside and outside the system
- Triad (customer, developer, tester) defines external behavior
- Decouple behavior specification and implementation

A Story – Speed Control

- As a driver, I want to set the speed for my car, and have it travel at that speed
Exercise
- Create a simple context diagram for a story in your backlog
- OR
- You have been chosen to develop a speed limit finder which uses speed signs to determine the speed limit
- Form a triad and create a simple context diagram
- OR
- You have been chosen to develop a speed limit finder which uses GPS location and map data to determine the speed limit

Possible Answer
- Speed Limit Finder

External Behavior
- Who wants a fast car?
- What behaviors do you desire for this car?
Behavior
How a system/component reacts to a stimulus (input, event, etc.)

**Requirement** → Required behavior of a system/component

External behavior can be described by
Use cases, requirements, acceptance tests, user acceptance test, scenarios etc.

---

**Scenario Template**
Scenario describes one behavior

**Given some state**
**When event or action occurs**
**Then new state or output**

---

**Behavior / Tests for Speed Control**
Setting speed causes speed to be close to set speed
Given car is active
When speed is set to 30 mph
Then car travels between 29 mph and 30 mph
TEST setting speed causes speed to be close to set speed
Given car is active
When speed is set to 30 mph
Then CHECK car travels between 29 mph and 30 mph

---

**Context**

Driver ➔ Car ➔ Movement

---
Behavior / Tests

Setting speed causes change to speed to occur rapidly
Given car is traveling at 20 mph
When speed is set to 30 mph
Then car reaches 29 mph within 3 seconds

TEST Setting speed causes change to speed to occur rapidly
Given car is traveling at 20 mph
When speed is set to 30 mph
Then CHECK car reaches 29 mph within 3 seconds

Alternatives for Describing Behavior

<table>
<thead>
<tr>
<th>Initial State</th>
<th>Given</th>
<th>Setup</th>
<th>Arrange</th>
<th>Assemble</th>
<th>Pre-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>When</td>
<td>Trigger</td>
<td>Act</td>
<td>Activate</td>
<td>Main course / exceptions / alternatives</td>
</tr>
<tr>
<td>Final State and/or Output</td>
<td>Then</td>
<td>Verify</td>
<td>Assert</td>
<td>Assert</td>
<td>Post-conditions</td>
</tr>
</tbody>
</table>

Requirements and Tests

Failing test is a behavior not yet met
Passing test is specification of the system behavior

Requirements and tests are inter-related
• You can’t have one without the other

Think Test First

No code goes in until the test goes on

Think Test-First

Don’t test code, code to the test
Exercise

- Create a scenario for the story from your backlog
- OR
- Create a scenario for the speed limit from signs
- OR
- Create a scenario for the speed limit from GPS

Possible Answers

Given car is active
When speed limit sign is viewed
Then speed limit is displayed

Given speed limit sign

```
SPEED LIMIT
35
```

When car views it
Then 35 mph is displayed

Possible Answers

Given speed limit for N 45.1, W 47.2 is 35 mph
When car is at N 45.1, W 47.2
Then 35 mph is displayed

Allocating Behavior

Synthesis versus Differentiation

“Within this process, every individual act of building is a process in which space gets differentiated. It is not a process of addition, in which preformed parts are combined to create a whole, but a process of unfolding, like the evolution of an embryo, in which the whole precedes the parts, and actually gives birth to them, by splitting”

Christopher W. Alexander

Synthesis

```
Bedroom
Dining Room
Living Room
Bedroom
```

```
Bedroom
Dining Room
Living Room
Bedroom
```
Differentiation

House

- Sleeping
- Living

Bedroom

Living Room

Bedroom

Dining Room

- Process is both
- Differentiate for cohesive sets of behaviors
- Synthesize from existing components with desired behaviors

Power of Three

- Three ways to do something
  So that tradeoffs become clearer
- Three people creating behavior specifications (the triad)
- Three people looking at code (pair plus one)

Internal Behavior

- External behavior – defined by the Triad
- Every internal component (e.g. function, class, microservice, etc.) helps produce external behavior
- Design ➔ Determine what internal components and their behavior required to meet external behavior
  - Look at three alternatives
  - Decide on the tradeoffs
  - Triad (user/caller, creator, tester) defines behavior of components
    - These are roles, not individuals

One Design for Fast Car

- Driver
- Accelerates
- Engine
- Transmission
- Wheels
- Movement
If You’re D’out, Try A Holistic Approach

Workflow

Workflow Example

Composite

Exercise

One Alternative Differentiation

Another Alternative Differentiation

© 2017-2019 Ken Pugh
Still Another Alternative Differentiation

Behavior / Test

Capture Image

Human interprets sign

Display speed limit

行为 / 测试

• Allocate behavior between components
  • All the way down (differentiate)
  • Until lowest level or existing component (synthesize)

Automation of Behavior Tests

Testing and behavior

• Testing ➔
  • Checking that desired behavior is provided by an implementation
• Regression testing ➔
  • Double checking that behavior has not changed when other behaviors introduced
• Behaviors with dependencies on slow/expensive/random components need test doubles / mocks

Test Doubles

Driver

Set Speed

Car

Drive at Speed

Test Doubles for Components

Tests

User Interface

User Interface Test Double

Processing

Processing Test Double

Persistence

Persistence Test Double

© 2017-2019 Ken Pugh
Automation

- Test automation is not something bolted on, but built in
- Use tests of behavior to design for automation
  - Identify appropriate test doubles

- Tests of behavior should be decoupled from implementation
  - Use "glue code" to connect test to production code
  - One test may be reused to connect to different levels
    e.g. UI, mid-tier, components

Testing Levels

Exercise

- What test doubles might you need for your own story?
- Or
- What test doubles might be needed for the speed limit finder story?

Possible Answer

- External Behavior
  - Video of driving
- Internal Behavior
  - Set of sign images
  - Display

Understanding Domain Term and Business Rule Behavior

Domain terms

- Domain terms – used in defining the domain
  - Domain terms often become classes
Domain Term Behavior

- Terms have behavior – e.g. what values they can have

<table>
<thead>
<tr>
<th>Speed</th>
<th>Valid</th>
<th>Direction</th>
<th>Valid</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>1</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>249</td>
<td>Yes</td>
<td>259</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>No</td>
<td>250</td>
<td>No</td>
<td>Wraps to 0</td>
</tr>
</tbody>
</table>

Business Rules

- Business rules
  - Usually specified by customer
  - Independent of implementation

- Business rules have behavior
  - The result they produce
  - Possible change in flow created by that result

Change in Speed

- Multiple ways to change speed
  - Mileage sensitive
  - Time sensitive

<table>
<thead>
<tr>
<th>Current speed</th>
<th>Desired speed</th>
<th>Distance</th>
<th>Goal</th>
<th>Deceleration rate?</th>
<th>Jerk Deceleration?</th>
<th>Start at distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 mph</td>
<td>30 mph</td>
<td>200 feet</td>
<td>time</td>
<td>5 feet/sec^2</td>
<td>2 feet/sec^3</td>
<td>0 feet</td>
</tr>
<tr>
<td>40 mph</td>
<td>30 mph</td>
<td>200 feet</td>
<td>mileage</td>
<td>10 feet/sec^2</td>
<td>5 feet/sec^3</td>
<td>100 feet</td>
</tr>
</tbody>
</table>

Exercise

- Write the business rule behavior test for a school zone speed limit sign
- Details follow
If You’re D’d out, Try A Holistic Approach

School Zone Context

Split the Behavior

One Possible Answer

Define behavior for quality attributes

Behavior is not just functional

<table>
<thead>
<tr>
<th>Time</th>
<th>Speed Limit</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:44 AM</td>
<td>45 mph</td>
<td>Speed limit for road</td>
</tr>
<tr>
<td>7:45 AM</td>
<td>35 mph</td>
<td>School zone applies</td>
</tr>
<tr>
<td>8:44 AM</td>
<td>35 mph</td>
<td>School zone applies</td>
</tr>
<tr>
<td>8:45 AM</td>
<td>45 mph</td>
<td>Speed limit for road</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change speed energy usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference in speed</td>
</tr>
<tr>
<td>10 mph</td>
</tr>
<tr>
<td>50 mph</td>
</tr>
</tbody>
</table>

- Behavior has quality attributes
- AKA Cross-functional
- AKA Non-functional
- Quality attributes may include
  - Performance
  - Reliability
  - What else?
**Exercise**

- What are the quality attributes for the behavior you have specified?

**State Dependent Behavior**

If there's time

**Signs identified**

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Percentage identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daylight clear</td>
<td>99.99%</td>
</tr>
<tr>
<td>Daylight rainy</td>
<td>99%</td>
</tr>
<tr>
<td>Dark clear</td>
<td>98%</td>
</tr>
<tr>
<td>Dark rainy</td>
<td>96%</td>
</tr>
</tbody>
</table>

**Components (e.g. Entities or objects) often:**

- Have states and go through transitions between states
- May behave differently based on what state they are in
  - See what condition their condition is in
  - Then act/transition based on events

**Model state-dependent behavior with State models**

- Speed controller responds differently depending on its state

**State transition table shows transitions between states**

- Table can be used in place of the diagram
- Shows non-applicable state/transition pairs (N/A)

<table>
<thead>
<tr>
<th>State/Event</th>
<th>Speed up</th>
<th>Too Slow</th>
<th>Too Fast</th>
<th>At Speed</th>
<th>At 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stopped</td>
<td>Accelerating</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Accelerating</td>
<td>N/A</td>
<td>N/A</td>
<td>Decelerating</td>
<td>Maintain Speed</td>
<td>N/A</td>
</tr>
<tr>
<td>Decelerating</td>
<td>N/A</td>
<td>Accelerating</td>
<td>N/A</td>
<td>Maintain Speed</td>
<td>Stopped</td>
</tr>
<tr>
<td>Maintain Speed</td>
<td>N/A</td>
<td>Accelerating</td>
<td>Decelerating</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Are we missing any transitions?
Exercise

• Is there a state-dependent behavior in your system?

Outcomes

• Explain the different contexts of an application
• Identify what are external and internal behaviors and how to specify them
• Create a shared understanding of the behavior of business rules and domain terms
• Minimize the amount of redundancy in describing behavior

It’s Not the Ending, But a Beginning

Go Forth and Behave Yourself

Think Behavior All the Way Down

behavior for event may cause
• Change to state
• An operation or output

Every state/event transition should have a test for it
– For the correct state transition
– For the correct response

Given state is Maintain Speed
When event Too Fast occurs
Then state becomes Decelerating
ask Brake to decelerate