Who are you?

- Testers, developers, others?
- Why are you here?

Objectives / Tests

- Understand a model for testing microservices
- Collaborate on creating microservice specifications with tests
- Create appropriate cross-functional tests for microservices
- Determine a strategy to test microservice interactions
- Explore ways to use logging
- Possibly apply model to your microservice

Ken Pugh

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

atdd@kenpugh.com
https://www.linkedin.com/in/kenpugh/
http://acceptancetestdrivendevelopment.com

Overall Rule

There are exceptions to every statement, except this one

© 2017-2019 Ken Pugh
2\textsuperscript{nd} Overall Rule

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

Perspective

See The Big Picture

Context, External Behavior, Interactions
Monolith versus Microservices

Example Domain

• Premium Member
• Nights – to achieve Premium Level
  • Staying in hotel
• Points – for Rewards
  • Achieve by staying in hotel
  • Using credit card
  • Doing promotion

Context
**Testing Microservices**

See It, Feel It, Touch It, Heal It, Explore It

---

**Context of Application**

- User
- System
- External Service
- Persistence Service

---

**Example Context**

- Clerk checks in/out guest
- Premium Member looks at points
- Credit Card Processor
- Reservation Database

---

**Collaboration helps automation**

---

**Behavior**

- Flow
- Given State
- When Input Event
- Then Output Event Change in State

---

**External Behavior (1)**

- Given a renter occupies a hotel room
- When the renter checks out
- Then a bill is produced with all charges including taxes
- And a transaction is issued to the credit card company
- And the room is not occupied

---

© 2017-2019 Ken Pugh
External Behavior (2)

- Given a renter who is a premium member occupies a hotel room
- When the renter checks out
- Then the renter is given night credits for each night
- And points for the total charges

- Given a premium member is logged on
- When they request their status history
- Then qualifying activity is displayed for every category

Microservice Attributes

- Self-contained business functionality
- Clear interfaces
- Independently deployable
- Independently scalable (benefit)

- A micro application

Monolith / Microservices

Microservice Context

Tests

Monolith Testing

User Interface

Processing

Persistence

User Interface Test Double

Processing Test Double

Persistence Test Double

Clerk

Checks out guest

Credit Card Processor

Reservation Service

Monolith Testing

See It, Feel It, Touch It, Heal It, Explore It

© 2017-2019 Ken Pugh
Example of Premium Member Context

- Updates nights
- Premium Member
- Checkout Service
- Points Updater
- Updates points
- Points / Nights Display

Example Context

- Hotel
  - Reserve a room
    - Compute taxes
    - May use premium points
  - Check in
  - Check out
    - Compute taxes
    - Record nights and points
    - May record use of points

Exercise

- Draw a context diagram for a microservice you are responsible for testing
- Or
- Draw a context diagram for a hotel tax service
  - Computes taxes for hotel stay

Example of Tax Service Context

- Checkout Service
- Computes taxes
- Tax Collector
- Pay Taxes
- Tax Updater
- Updates tax rates
- External Service

Interfaces

Monolith

Microservice

Testing in an environment

© 2017-2019 Ken Pugh
Three Laws of Interfaces

1. An implementation of an interface shall do what the interface defines (See It)
2. An implementation shall do no harm (use excessive memory, hold locks, etc.) (Feel It)
3. If an implementation is unable to perform its responsibilities, it shall notify someone (Heal It)

Model

Information Model

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Date Of Stay</th>
<th>Number Nights</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1/1/2019</td>
<td>1</td>
</tr>
</tbody>
</table>

```
MemberID, DateOfStay, NumberNights,
M123, 1/1/2019, 1
<DateOfStay> 1/1/2019 </DateOfStay>
<DateOfStay> 1/1/2019 </DateOfStay>
{ "member_id" : "M123", 
  "date_of_stay" : "2019-01-01", 
  "number_nights" : 1 }
```
Information Model (2)

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Date Of Stay</th>
<th>Number Nights</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1/1/2019</td>
<td>1</td>
</tr>
<tr>
<td>M123</td>
<td>2/2/2019</td>
<td>2</td>
</tr>
</tbody>
</table>

```json
{
  "member_id": "M123",
  "date_of_stay": "2019-01-01",
  "number_nights": 1
},
{
  "member_id": "M123",
  "date_of_stay": "2019-02-02",
  "number_nights": 2
}
```

Information Model (3)

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Date Of Stay</th>
<th>Number Nights</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1/1/2019</td>
<td>1</td>
</tr>
</tbody>
</table>

```

Behavior

Behavior for Scenario (Flow)

```

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Date Of Stay</th>
<th>Number Nights</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1/1/2019</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Example of Behavior

• Member status updated on checkout
  • Given Member status is
  • When checkout occurs
  • Then Member status is

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Number of Nights</th>
<th>Current Year</th>
<th>Total Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>11</td>
<td></td>
<td>100,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of Nights</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/1/2019</td>
<td>2</td>
<td>$100.00</td>
</tr>
</tbody>
</table>

Business Rule
10 points for every dollar spent
Exercise

- Write up a scenario/test for a microservice you are responsible for testing
- Or
- Write up a scenario/test for a hotel tax service

### Example of Taxes

- **Given Tax Rates are**
<table>
<thead>
<tr>
<th>Zip Code</th>
<th>State Occupancy Tax Rate</th>
<th>County Occupancy Tax Rate</th>
<th>City Occupancy Tax Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>27701</td>
<td>0.07</td>
<td>0.02</td>
<td>0.01</td>
</tr>
</tbody>
</table>

- **When taxes are computed**

<table>
<thead>
<tr>
<th>Zip Code</th>
<th>Room Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>27701</td>
<td>100.00</td>
</tr>
</tbody>
</table>

- **Then taxes are**

<table>
<thead>
<tr>
<th>State Occupancy Tax</th>
<th>County Occupancy Tax</th>
<th>City Occupancy Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.00</td>
<td>2.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

### Breaking Down (1)

- **Given Member status is**

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Number of Nights</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>12</td>
<td>100,000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>

- **When checkout occurs**

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of Nights</th>
<th>Points</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/1/2019</td>
<td>1</td>
<td>100,000</td>
<td>101,000</td>
</tr>
</tbody>
</table>

- **Then Point Transaction is created**

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>

### Breaking Down (2)

- **Add Point Transaction**

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>1000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>

- **Given current Point Total is**

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>100,000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>

### Another Example of Behavior

- **Current member status displayed**

<table>
<thead>
<tr>
<th>Member ID</th>
<th>Number of Nights</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>12</td>
<td>101,000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>

- **When Member requests display**

<table>
<thead>
<tr>
<th>Number of Nights</th>
<th>Points</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>101,000</td>
<td>7/1/2019</td>
<td>Hotel Stay</td>
</tr>
</tbody>
</table>
Testing Microservices
See It, Feel It, Touch It, Heal It, Explore It

**Sequence / Workflow**

Given  When  Then

- Member status updated on checkout
- Member status display
- Points used to reserve room

**Behavior for Business Rule**

- Setup (Given)
- Trigger (When)
- Assert (Then)
- Input Values
- Business Rule
- Expected Results

**Test for Business Rule**

- Setup (Given)
- Trigger (When)
- Assert (Then)
- Input Values
- Business Rule
- Actual Results
- Expected Results

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>value</td>
<td>result</td>
</tr>
</tbody>
</table>

**Example of Business Rule**

- **Member Premium Level Business Rule**
  - Number of Nights Current Year | Premium Level
  - 10  | Silver
  - 25  | Gold
  - 50  | Platinum
  - 100 | Diamonds

- **Tests of Business Rule**
  - Number of Nights Current Year | Premium Level?
  - 10  | Gold
  - 20  | Silver
  - 25  | Gold
  - No  | No

**Interface (Domain) Terms**

<table>
<thead>
<tr>
<th>Points – Rules and Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category Values For Points</td>
</tr>
<tr>
<td>- Enumeration</td>
</tr>
<tr>
<td>- Hotel Stay</td>
</tr>
<tr>
<td>- Credit Card</td>
</tr>
<tr>
<td>- Yearly Bonus</td>
</tr>
<tr>
<td>- Redemption</td>
</tr>
<tr>
<td>- Redemption Cancellation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Valid?</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>No</td>
<td>0</td>
<td>200000</td>
<td>Integer</td>
</tr>
<tr>
<td>0</td>
<td>Yes</td>
<td>0</td>
<td>200000</td>
<td>Integer</td>
</tr>
<tr>
<td>200001</td>
<td>No</td>
<td>0</td>
<td>200001</td>
<td>Integer</td>
</tr>
<tr>
<td>1.23</td>
<td>No</td>
<td>0</td>
<td>200001</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Invalid Values should produce error or default result.

© 2017-2019 Ken Pugh
Exercise

- Is there a business rule or an interface domain term in your microservice?
- If so, write up a behavior for it

Possible Answers

<table>
<thead>
<tr>
<th>Tax Rate</th>
<th>Room Charge</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00.00</td>
<td>100.00</td>
<td>Yes</td>
</tr>
<tr>
<td>.07</td>
<td>100.07</td>
<td>Yes</td>
</tr>
<tr>
<td>.07</td>
<td>100.08</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Zip Code Validation

<table>
<thead>
<tr>
<th>Zip Code</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>27701</td>
<td>Yes</td>
</tr>
<tr>
<td>27706</td>
<td>Yes</td>
</tr>
<tr>
<td>15754</td>
<td>No</td>
</tr>
</tbody>
</table>

Tax Rates Validation

<table>
<thead>
<tr>
<th>Tax Rate</th>
<th>Room Charge</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>-01</td>
<td>100.00</td>
<td>No</td>
</tr>
<tr>
<td>00.00</td>
<td>100.07</td>
<td>Yes</td>
</tr>
<tr>
<td>.30</td>
<td>100.08</td>
<td>No</td>
</tr>
</tbody>
</table>

Round Off

<table>
<thead>
<tr>
<th>Tax Rate</th>
<th>Room Charge</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>.07</td>
<td>100.07</td>
<td>Yes</td>
</tr>
<tr>
<td>.07</td>
<td>100.08</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Interactions

Sequenced

Orchestrated

Testing Communication

Stimulus/Response
Testing Microservices
See It, Feel It, Touch It, Heal It, Explore It

Events

Event Examples

• Premium Members
  • Add points
  • Use points
    • On checkout if stay is on points
• Event
  • Checkout Event

<table>
<thead>
<tr>
<th>Member ID</th>
<th>0123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>7/2/2019</td>
</tr>
<tr>
<td>Number of Nights</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>$50</td>
</tr>
<tr>
<td>Points Used</td>
<td>$10,000</td>
</tr>
</tbody>
</table>

© 2017-2019 Ken Pugh
Testing Microservices
See It, Feel It, Touch It, Heal It, Explore It

Event Example (2)

• But what other events should Premium Member subscribe to?
  
  • Reserve use of points
  
<table>
<thead>
<tr>
<th>Reservation ID</th>
<th>Points Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>R123</td>
<td>20,000</td>
</tr>
</tbody>
</table>

• Cancel use of points
  
<table>
<thead>
<tr>
<th>Reservation ID</th>
<th>Points Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>R123</td>
<td>20,000</td>
</tr>
</tbody>
</table>

Example of Event Stream

• Given Member status is
  
<table>
<thead>
<tr>
<th>Member ID</th>
<th>Number of Nights Current Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>0</td>
</tr>
</tbody>
</table>

• When events occur
  
<table>
<thead>
<tr>
<th>Date</th>
<th>Number of Nights</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1/2019</td>
<td>5</td>
</tr>
<tr>
<td>3/1/2019</td>
<td>1</td>
</tr>
<tr>
<td>4/1/2019</td>
<td>5</td>
</tr>
</tbody>
</table>

• Then Member status is
  
<table>
<thead>
<tr>
<th>Member ID</th>
<th>Number of Nights Current Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>12</td>
</tr>
</tbody>
</table>

Another Example of Event Stream

• Given Point Total is
  
<table>
<thead>
<tr>
<th>Member ID</th>
<th>Point Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>100,000</td>
</tr>
</tbody>
</table>

• When events occur
  
<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2/2019</td>
<td>Credit Card Purchases</td>
<td>1000</td>
</tr>
<tr>
<td>3/1/2019</td>
<td>Room</td>
<td>500</td>
</tr>
<tr>
<td>4/1/2019</td>
<td>Redemption</td>
<td>20,000</td>
</tr>
</tbody>
</table>

• Then Point Total is
  
<table>
<thead>
<tr>
<th>Member ID</th>
<th>Point Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>M123</td>
<td>81,500</td>
</tr>
</tbody>
</table>

Question

• How do the microservices in your context collaborate?

Feel It
Behavior for cross-functional aspects of microservices documented in tests

Agile Testing Matrix

<table>
<thead>
<tr>
<th>Technology Facing</th>
<th>Business Facing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapted from</td>
<td></td>
</tr>
<tr>
<td>Janet Gregory</td>
<td></td>
</tr>
<tr>
<td>Lisa Crispin</td>
<td></td>
</tr>
</tbody>
</table>

© 2017-2019 Ken Pugh
Feel It

- Quality attributes
  - AKA Cross-functional requirements
  - AKA Non-functional requirements

- What quality attributes do you currently test in your systems?
  - Service Level Agreement (SLA)

Service Level Agreement

<table>
<thead>
<tr>
<th>Total latency</th>
<th>5 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput</td>
<td>50000 requests/second</td>
</tr>
<tr>
<td>Processing time</td>
<td>20 microseconds</td>
</tr>
</tbody>
</table>

Resource Agreement

<table>
<thead>
<tr>
<th>Memory usage</th>
<th>&lt; 100 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOP per call</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Testing Environment

- Environment (framework / libraries / etc.) can provide aspects of quality attributes
  - E.g. Increase instances

- Are you testing the environment, the configuration, or the service itself?

Example of Idempotent Test

- Given Point Total is
  - Member ID | Point Total
  - M123 | 100,000

- When duplicate transactions occur
  - Date | Type | Amount | Transaction ID
  - 1/2/2019 | Credit Card Purchases | 1000 | T456
  - 1/2/2019 | Credit Card Purchases | 1000 | T456
  - 1/2/2019 | Credit Card Purchases | 1000 | T456

- Then Point Total is
  - Member ID | Point Total
  - M123 | 101,000

Idempotency

- Same outcome if
  - Request received multiple times
  - Response received multiple times

- One way is to add ID
  - Event ID
  - Transaction ID
  - Correlation ID

Logging
Testing Microservices
See It, Feel It, Touch It, Heal It, Explore It

Logging

- Log events with:
  - Timestamp
  - ID (Event, Correlation, etc.)
  - Source
  - Service
  - Operational location
- Log entries include:
  - Requests / responses
  - Significant internal events (test points)

Discussion

- What else might be on an SLA?

Touch It
Test for interactions of microservices include injection or simulation of faults

What’s Going On?

We're sorry but we can't tell you what your reservations are at the current time. Call 919 555-1234 for an explanation

Testing Communication

Injecting Faults

- Test Double could
  - Return unavailable
  - Return after a long time
  - Return with dependency issue
  - Return other potential errors
Exercise

- Would the microservice you are testing require test doubles or should it have a test double?
- Or
- What should a test double for taxes do?

Example of Taxes

- Return tax calculated at fixed rate, regardless of jurisdiction

Heal It

Checking of logs of microservice interactions can be used to determine failure causes

Recording the Symptoms

- Service Logging
  - Record request/responses of services
  - Particularly error responses
  - Record significant events (test points) inside a service
- Analyze logging for
  - Service Level Agreement
  - What is "normal"
    - Error rates
    - Performance
  - Look for abnormal
    - What is significantly different

“Testing in Production”

- Synthetic (AKA Fake) transactions or synthetic users
  - Perform operations to test workflows in production
  - E.G. Reserve room and then unreserve it

Discussion

- Are there any significant events in your service that should be recorded?
- What level of errors is acceptable for your system?
- Can you use synthetic transactions?
- Testing is about tradeoffs - effort versus benefit
  - How much testing should you do on complete system?
Explore It
Testing going beyond described behaviors.

Exploratory Testing
- Usually done on application as a whole
- Could be done on individual service
- Could be done on multiple services

Exploratory Testing
- Usually done on application as a whole
- Could be done on individual service
- Could be done on multiple services

Exercise
- Anything you want to explore with taxes?

Objectives / Tests Review
- Understand a model for testing microservices
- Collaborate on creating microservice specifications with tests
- Create appropriate cross-functional tests for microservices
- Determine a strategy to test microservice interactions
- Explore ways to use logging
- See It
- Feel It
- Touch It
- Heal It
- Explore IT

Go Forth and See, Feel, Touch, Heal, Explore Your Microservices

Ken Pugh
atdd@kenpugh.com
https://www.linkedin.com/in/kenpugh/
http://acceptancetestdrivendevelopment.com

© 2017-2019 Ken Pugh