Use Tables to Drive out Ambiguity / Redundancy, Discover Scenarios, and Solve World Hunger

Ken Pugh

Ken Pugh

ken.pugh@pugh-killeen.com

- ATDD/BDD, TDD, OOA&D, Design Patterns, Lean, Scrum, SAFe
- Over 2/5 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*

No code goes in, till the test goes on.
A journey of two thousand miles begins with a single step.

Outline

- **Introduction to Acceptance Test Driven Development / Behavior Driven Development**
- **Tables**
  - **Effective form of communication**
  - **For acceptance tests / requirements / other**
- **Tables can:**
  - **Help analyze scenarios**
  - **Reduce redundant scenarios and tests**
  - **Discover missing scenarios**
  - **Create domain specific language (DSL)**
- **Show tables in Fit and Gherkin syntax**

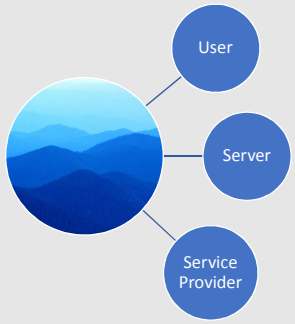
Overall Rule

- **There are exceptions to every statement, except this one**

Introduction to ATDD/BDD

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**

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)**

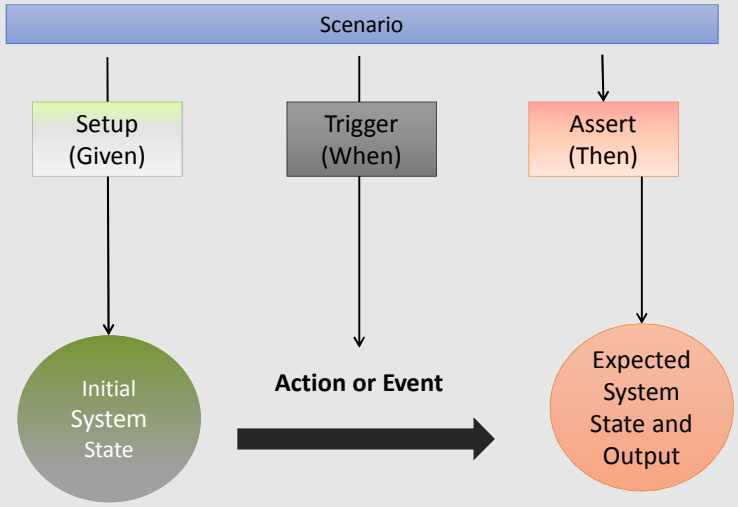
Why?

- **Rework Down from 60% to 20%**
- **Little Room for Miscommunication**
- **Workflows Working First Time**
- **Getting Business Rules Right**
- **Crisp Visible Story Completion Criteria**
- **Tighter Cross-Functional Team Integration**

Requirements and Tests

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

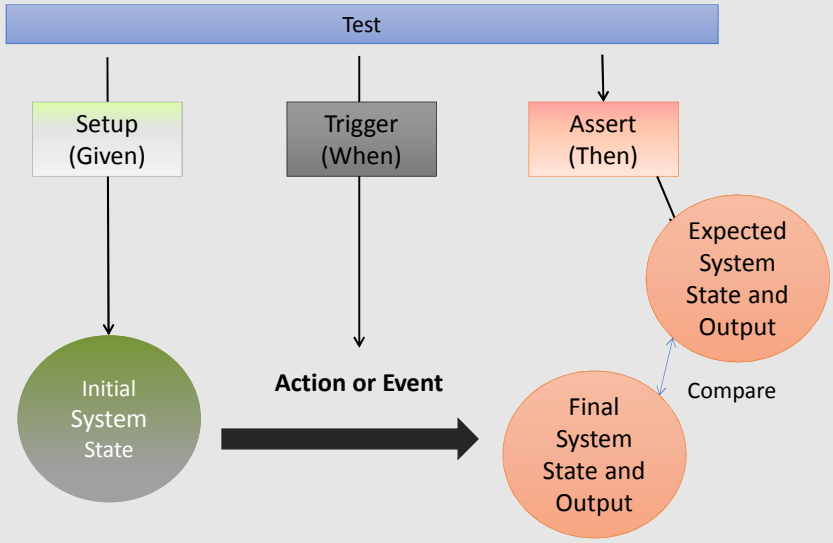
Scenario Flow



Example of Scenario

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

Test Flow



Example of Test Against Scenario

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

Acceptance Test Example

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%.

What discount for a Good Customer and $50.01 Order Total?

Given / When / Then

```
Given total is 50.01 and rating is Good
When I compute discount
Then percent is 1

Given total is 10.00 and rating is Good
When I compute discount
Then percent is 0

Given total is 10.01 and rating is Good
When I compute discount
Then percent is 1
```

Avoid Copy and Paste

In requirements, tests, code, documentation

Business Rule Table = Test

Discount		
Order total	**Customer rating**	**Discount percentage?**
$50.01	Good	1%
$10.00	Good	0%
$10.01	Good	1%
$.01	Excellent	1%
$50.00	Excellent	1%
$50.01	Excellent	5%

Ways To Implement Test

- **Testing script**
- **Unit test 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 $50.01**
  - **Complete order**
  - **Check it shows a $.50 discount**
- **Repeat for other five cases**

XUnit Test

```
class TestCase  {
   Void 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(1, o.computeDiscount(50.0, Excellent));
       assertEquals(5, o.computeDiscount(50.01, Excellent));
       }
   }
```

Fit (Table = Test)

Discount		
Order total	**Customer rating**	**Discount percentage?**
10.00	Good	0
10.01	Good	1
50.01	Good	1
.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

Fit Test

Discount		
Order total	**Customer rating**	**Discount percentage?**
10.00	Good	0
10.01	Good	1
50.01	Good	Expected 1 Actual 5
.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

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                 |
```

Table As Requirement

Requirement

Discount Rule		
Customer Rating	**Order Total**	**Discount Percentage**
Good	<= $10.00	0%
	Otherwise	1%
Excellent	Any	1%
	> $50.00	5%

Requirement restated

Discount Rule		
Customer Rating	**Order Total**	**Discount Percentage**
Good	<= $10.00	0%
	> $10.00	1%
Excellent	<= $50.00	1%
	> $50.00	5%

Calculation Table (Business Rule)

Title			
Input Name 1	**Input Name 2**	**Result Name?**	**Notes**
Value for input 1	Value for input 2	Expected output	description

Discount		
Order total	**Customer rating**	**Discount percentage?**
10.00	Good	0
10.01	Good	1
50.01	Good	1
.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

Domain Terms

- **Acceptance tests should use business domain language**
- **Make up tables/tests to elaborate terms**
- **Use tables to discover missing scenarios**

CustomerRating
Good
Excellent
????

What about other Customer Ratings?

Splitting Tables

Splitting Tables

- **Wide or high tables are hard to read**
- **Probably a hidden business rule (or sub-business rule)**
- **Splitting is similar to using Karnaugh maps for simplifying boolean logic**

A Story

- **If Customer Level is Silver with 10000 miles or more or Customer Level is Gold with 20000 miles or Customer Level is Platinum with 30000 miles or more**
- **Then give 1000 bonus miles and 100 elite qualifying miles**

CustomerLevel
None
Silver
Gold
Platinum

Original Table

CustomerLevel	Miles	Bonus Miles?	Elite Qualifying Miles ?
Silver	10000	1000	100
Gold	20000	1000	100
Platinum	30000	1000	100
Silver	9999	0	0
Gold	19999	0	0
Platinum	29999	0	0

Splitting Tables

- **Split policy from result**

CustomerLevel	Miles	Special Offer?
Silver	10000	Yes
Gold	20000	Yes
Platinum	30000	Yes
Silver	9999	No
Gold	19999	No
Platinum	29999	No

Special Offer	Bonus Miles?	Elite Qualifying Miles?
Yes	1000	100
No	0	0

Exercise

Action	Giver Has Email	Receiver Has Email	Send via US Mail?	Form ?
Transfer	Yes	Yes	No	F-7421
Transfer	Yes	No	Yes	F-742
Transfer	No	Yes	Yes	F-742
Transfer	No	No	Yes	F-742
Propose	Yes	Yes	No	K-481
Propose	Yes	No	Yes	K-48
Propose	No	Yes	Yes	K-48
Propose	No	No	Yes	K-48

One Answer

Action	Email For Both	Send via US Mail?	Form ?
Transfer	Yes	No	F-7421
Transfer	No	Yes	F-742
Propose	Yes	No	K-481
Propose	No	Yes	K-48

Giver Has Email	Receiver Has Email	Email For Both?
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	No	No

Tables in a Flow

More Table Uses

- **Data Table (Persistence or Interface Data)**
- **Action Table (User Input)**

Data Table (Persistence or Interface Data)

- **Exists (or should exist) – for Given and Then**

Title Data	
Value Name 1	**Value Name 2**
Value for 1	Value for 2
Another value for 1	Another value for 2

Data Table Example

Customer Data	
Name	**ID**
James	007
Maxwell	86

Given (e.g. setup requires these customers)
or
Then (e.g. after adding Customer)

Action Table (User Input)

- ***Enter*** **enters data into an entry field**
- ***Execute*** **initiates a process, such as a Submit button on a dialog box**
- **Equivalent to method call with parameters**

Action Name		
Enter	**Value Name 1**	Value for 1
Enter	**Value Name 2**	Value for 2
Execute	**Submit**	

Action Table Example

Check Out CD		
Enter	Customer ID	007
Enter	CD ID	CD2
Execute	Rent	

Given / When / Then Example

- **Story: Rent a CD to a customer**
- **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 Flow Test (1)

Check Out CD

- **Given Customer has ID**

Customer Data	
Name	**ID**
James	007

Data Tables

and CD has ID

and CD is not currently rented

CD Data		
ID	**Title**	**Rented**
CD2	Beatles Greatest Hits	No

Full Example Flow Test (2)

- **When a clerk checks out a CD:**

Check Out CD		
Enter	**Customer ID**	007
Enter	**CD ID**	CD2
Execute	**Rent**	

Action Table

Full Example Flow Test (3)

- **Then the CD is recorded as rented**

Data Tables

CD Data			
ID	**Title**	**Rented**	**Customer ID**
CD2	Beatles Greatest Hits	Yes	007

and a rental contract is printed:

Rental Contract			
Customer ID	**Customer Name**	**CD ID**	**CD Title**
007	James	CD2	Beatles Greatest Hits

Anything else on the contract?

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:**

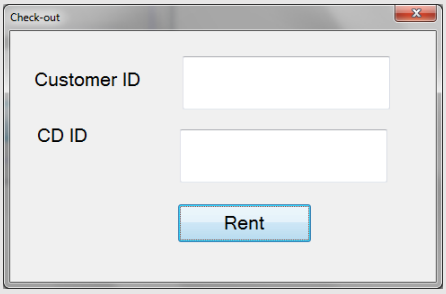
Today
1/1/2014

- **Then a rental contract is printed:**

Rental Contract					
Customer ID	**Customer Name**	**CD ID**	**CD Title**	**Due**	**Fee**
007	James	CD2	Beatles Greatest Hits	1/3/2014	$3

The Action

- **Can drive a GUI**



- **Or a method**

```
Rent(CustomerID aCustomer, CDID aCD);
```

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

Finding Scenarios

- **Look at existing given /when**
  - **Check different values and see if different results**
- **Example of different givens:**

CD Data			
ID	**Title**	**Rented**	**Customer ID**
CD2	Beatles Greatest Hits	No	

CD Data			
ID	**Title**	**Rented**	**Customer ID**
CD2	Beatles Greatest Hits	Yes	86

CD Data			
ID	**Title**	**Rented**	**Customer ID**
CD2	Beatles Greatest Hits	Yes	007

Gherkin with Tables

```
Scenario: Normal Check Out
  Given customer exists:
       |ID | Name |
       |007| James|
  And CD exists:
       |ID  | Title                 | Rented |
       |CD2 | Beatles Greatest Hits | No     |
  When checkout occurs:
       |Field      | Value |
       }CD         } CD2   |
       |Customer   | 007   |
  Then CD is now:
       |ID  | Title                 | Rented | Customer ID |
       |CD2 | Beatles Greatest Hits | Yes    | 007         |
  And rental contract produced:
       |Customer ID|Customer Name|CD ID |CD Title             |
       |007        |James        |CD2   |Beatles Greatest Hits |
```

Gherkin Without Tables

```
Scenario: Normal Check Out
  Given Customer with ID 007 with Name James
  And CD with ID CD2 Title "Beatles Greatest Hits"
       and Not Rented
  When CD CD2 is checked out to Customer 007
  Then CD CD2 is recorded as Rented to Customer 007
  And a rental contract produced with Customer ID 007
     Customer Name James CD ID CD2 CD Title
     "Beatles Greatest Hits"
```

Questions

- **Which is easier to parse for domain terms?**
- **If another value is added, which is easier to maintain?**

Domain Language Revisited

- **Many terms imply business rules for validation**
  - **E.g.: CD ID, Customer ID**
- **So each term should have its own table**
  - **Example: CD IDs start with CD and followed by digits**
  - **Customer ID contains only digits**

CDID		
Value	**Valid?**	**Notes**
CD2	Yes	
CDA73	No	CD plus numeric characters
C22	No	Must begin with CD

CustomerID		
Value	**Valid?**	**Notes**
2	Yes	
C	No	Non-numeric

Example with Types

- **When a clerk checks out a CD:**

Check Out CD			**Validation**
Enter	**Customer ID**	007	**CustomerID**
Enter	**CD ID**	CD2	**CDID**
Execute	**Rent**		

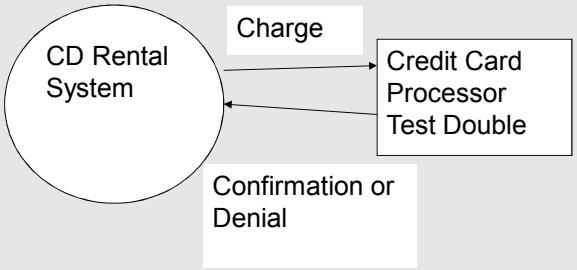
More Domain Terms

Dollar		
Value	**Valid?**	**Notes**
10.00	Yes	
10.001	No	Maximum two decimal digits

CreditCardAccountNumber			
Number	**Card Issuer**	**Valid?**	**Notes**
4012888888881881	Visa	Yes	
40128888888818812	Visa	No	Too many digits
4012888888881882	Visa	No	Bad check digit

CreditCard Issuer
Visa
Mastercard
Discover

Example of External Service



Credit Card Transaction			
Issuer	**Number**	**Expires**	**Amount**
Visa	4004440000000019	01/2020	3.00

Transaction Receipt		
Transaction ID	**Result**	**Amount**
123456789012345	Accepted	3.00

Tables with Types

Credit Card Transaction			
Issuer	**Number**	**Expires**	**Amount**
CreditCard Issuer	**CreditCardAccount Number**	**MonthYear**	**Dollar**
Visa	4004440000000019	01/2020	3.00

Transaction Receipt		
ID	**Result**	**Amount**
TransactionID	**TransactionResults**	**Dollar**
123456789012345	Accepted	3.00

Tables and Classes

Credit Card Transaction			
Issuer	**Number**	**Expires**	**Amount**
CreditCard Issuer	**CreditCardAccount Number**	**MonthYear**	**Dollar**
Visa	4004440000000019	01/2020	3.00

```
class CreditCardTransaction
   {
   CreditCardIssuer issuer;
   CreditCardAccountNumber number;
   MonthYear expires;
   Dollar amount;
   };
```

Calculation Table Revised

Discount		
Order total	**Customer rating**	**Discount percentage?**
Dollar	**CustomerRatingType**	**Percentage**
10.00	Good	0
10.01	Good	1

```
Percentage discountPercentage(Dollar orderTotal,
    CustomerRatingType customerRating);
// or
class DiscountCriteria {
    Dollar orderTotal;
    CustomerRatingType customerRating;
    }
Percentage discountPercentage(DiscountCriteria criteria);
```

Data

Table for Configuration

- **Is this a specification, test, or implementation?**

CustomerLevel	Eligible For Miles	Eligible For Free Drinks	Boarding Group
None			6
Silver	X		3
Gold	X		2
Platinum	X	X	1

Data as XML

```
<person>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <age>25</age>
    <address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
    </address>
    <phoneNumbers>
        <phoneNumber>
            <type>home</type>
            <number>
                <areacode>212</areacode>
                <exchange>555</exchange>
                <extension>1234</extension>
              </number>
         </phoneNumber>
         <phoneNumber>
             <type>fax</type>
       ….
         </phoneNumbers>
</person>
```

Data as Vertical Table

Person	**First Name**			John	
	Last Name			Smith	
	Age			25	
	Address	**Street Address**		21 2nd Street	
		City		New York	
		State		NY	
		Postal Code		10021	
	Phone Number	**Type**		Home	Fax
		Number	**Area Code**	212	646
			Exchange	555	555
			Extension	1234	4567

Data as Horizontal Table

Person										
First Name	**Last Name**	**Age**	**Address**				**Phone Number**			
			Street Address	**City**	**State**	**Postal Code**	**Type**	**Number**		
								Area Code	**Exchange**	**Extension**
John	Smith	25	21 2nd Street	New York	NY	10021	Home	212	555	1234
							Fax	646	555	4567

Separation Again

Complex Business Rule

A business rule determines whether a user is allowed to perform a certain operation
Fields may contain values or be blank
DNC = Do Not Care what is the value

Field One	Field Two	Field Three	Field Four	Result ?
> 20	< 50 or blank	>=100	Y	Allow
Otherwise or blank	DNC	DNC	DNC	Disallow
DNC	Otherwise	DNC	DNC	Disallow
DNC	DNC	Otherwise or blank	DNC	Disallow
DNC	DNC	DNC	N or blank	Disallow

Complex Business Rule Simplified

Field One	Result?
> 20	Allow
Blank	Disallow
Otherwise	Disallow

Field Three	Result ?
>=100	Allow
Blank	Disallow
Otherwise	Disallow

Field Two	Result?
< 50	Allow
Blank	Allow
Otherwise	Disallow

Field Four	Result?
Y	Allow
N	Disallow
Blank	Disallow
Otherwise	??

Field One	Field Two	Field Three	Field Four	Result ?
Allow	Allow	Allow	Allow	Allow
Disallow	DNC	DNC	DNC	Disallow
DNC	Disallow	DNC	DNC	Disallow
DNC	DNC	Disallow	DNC	Disallow
DNC	DNC	DNC	Disallow	Disallow

Use Alternate Tables

Create tables in alternative forms, if standard not appropriate

Sudoku Puzzle								
1			4			7		
	2			5			8	
		3			6			9
4			7			1		
	5			8			2	
		6			9			3
7			1			4		
	8			2			5	
		9			3			6

Sudoku Solution								
1	6	5	4	9	8	7	3	2
9	2	4	3	5	7	6	8	1
8	7	3	2	1	6	5	4	9
4	9	8	7	3	2	1	6	5
3	5	7	6	8	1	9	2	4
2	1	6	5	4	9	8	7	3
7	3	2	1	6	5	4	9	8
6	8	1	9	2	4	3	5	7
5	4	9	8	7	3	2	1	6

Required Performance		
Puzzle Type	**Time to Solve?**	**Notes**
Easy	.1 seconds	See easy determination
Impossible	1 second	No solution

Not an Ending, But a Beginning

Review

- **Tables are an effective communication form**
- **Tables can:**
  - **Help analyze scenarios**
  - **Reduce redundant tests**
  - **Discover missing scenarios**
  - **Create domain specific language (DSL)**

Solve World Hunger

- **3 June 2008, Rome**
  - **FAO Director-General Jacques Diouf**
  - **US$30 billion a year to enable 862 million people to be not hungry**
- **Software Development**
  - **$200 billion dollars a year (or more)**
  - **Save just 15%, world hunger is solved**
- **Redundancy Jar**

Go Forth and Become Table Creators

Thank you