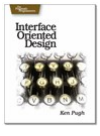
Ken Pugh

ken.pugh
@netobjectives.com



- Fellow Consultant
- OOA&D, Design Patterns, Lean, Scrum, Test-Driven Development
- 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.

Overall Rule

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

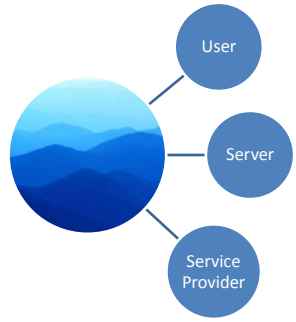
Outline

- Introduction
- Acceptance Test Examples
- Test Anatomy
- Supplemental

Introduction

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%
- 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

  (See ATTD book for details)

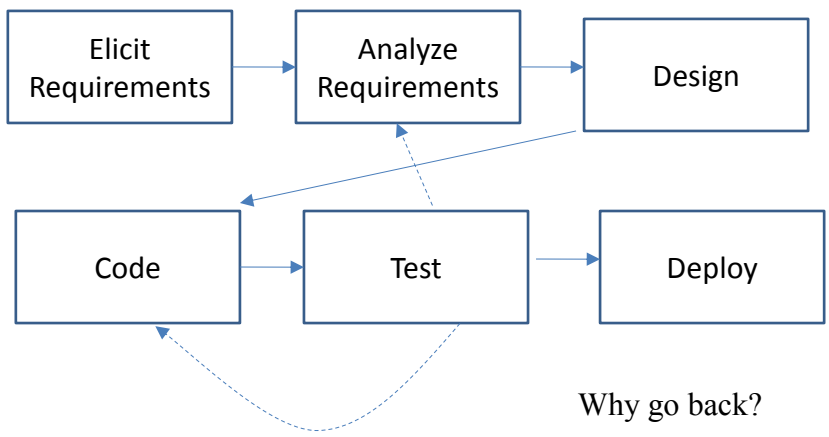
Requirements and Tests

- 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

The Team

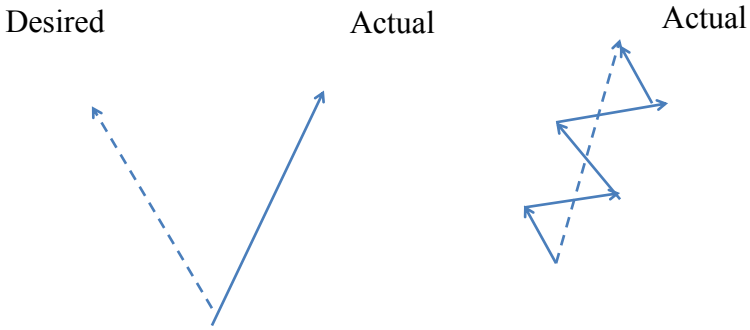
- Customer Unit - develops the requirements
  - Product Owner
  - Business analysts
  - Users
  - Quality Assurance
- Developer / Tester Unit – develops the implementation
  - Programmers
  - Testers / Quality Assurance
- The Triad
  - Customer, developer, tester

Value Stream Map – Classical Development

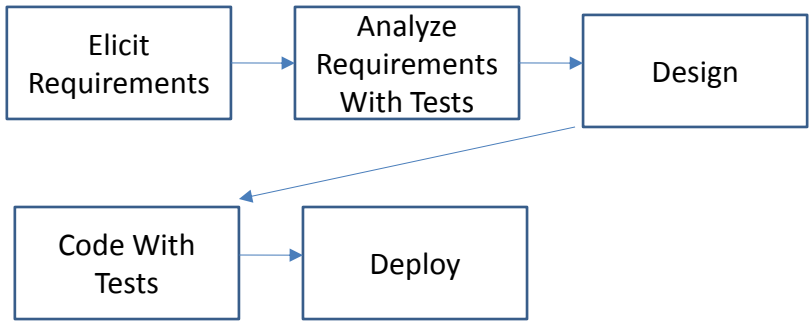


Why Mistakes?

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



Value Stream Map – Agile Development



Move Testing Forward

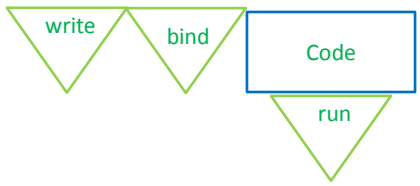
- Two types of testing
  - Attempting to find defects – is WA
  - Attempting to prevent defects – is ES
- When are defects found?
  - Prevention is just early detection

Steps

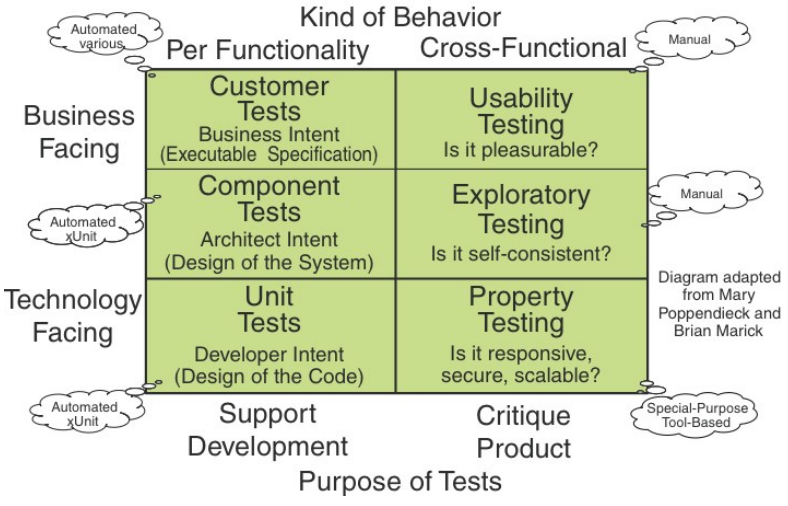
- Author the tests (write)
  - Customer, tester, developer together
- Connect tests to system (bind)
  - Developer unit
- Run the tests (run)
  - Developers, testers, customers
  - Automated – part of build

Usually the Case

A Better Way



Testing Strategies



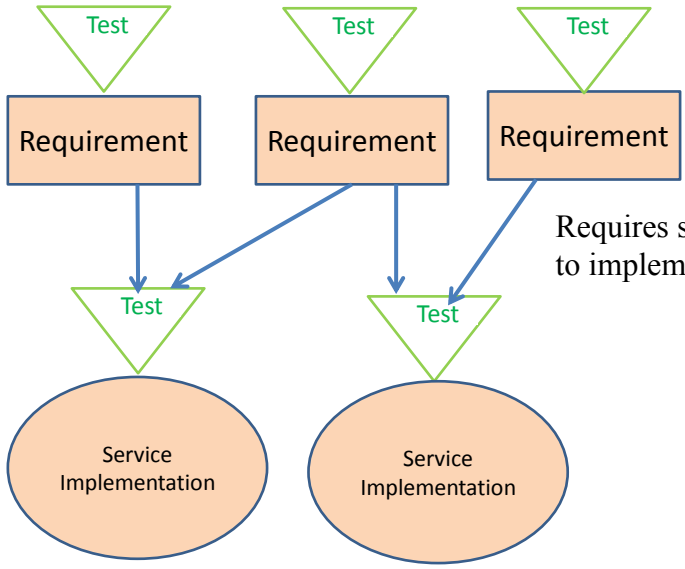
Meszaros, *XUNIT TEST PATTERNS: REFACTORING TEST CODE*, Fig 6.1 "Purpose of Tests" p. 51, © 2007 Pearson Education, Inc

Another Look at Testing Strategies

Focus of the tests (not exclusive)

	Customer Tests	**Exploratory Testing**
	Expected Results	**Unexpected Results**
Valid Input	Specified effects	Unspecified effects
Invalid input	Specified error message	Any side effects

Tests all the way down



ATDD 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%.

Business Rule 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%

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

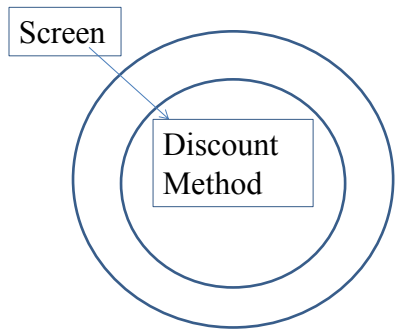
Program interface

- Create a command line or graphic user interface

Discount Percentage Screen

Customer Type: **Good**
Order Total: **10.01**

Percentage: **1 %**




```
C:>DiscountPercentage Good 10.01
Percentage: 1%
```


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(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

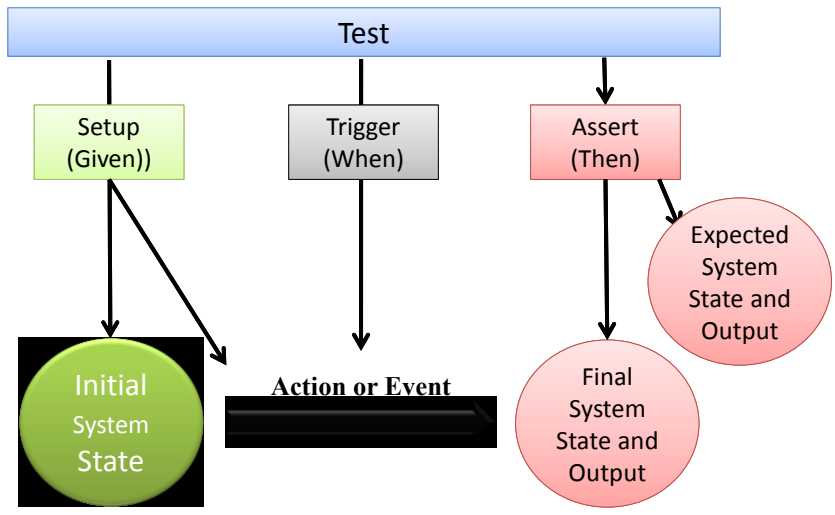
Discount		
Order total	**Customer rating**	**Discount percentage?**
10.00	Good	0
10.01	Good	1
50.01	Good	
.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

Test Anatomy

31

24 October 2014

Test Flow



32

Net Objectives

24 October 2014

Recap

When

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%

Given Then

User Story Examples

- As the clerk, I want to check out a CD for a customer.
- As the clerk, I want to check in a CD for a customer.

Use Case Example – Part One

- **Name**—Check Out CD.
- **Description**—Check out a CD for a customer.
- **Actor**—Clerk.
- **Stakeholders** – Inventory Manager, Security
- **Pre-conditions**—The customer has an identification. The CD has an identity.
- **Post-conditions**—The CD is recorded as rented. The rental contract is printed.
- **Main Course:**
  1. The clerk enters the customer identification and CD identifier into the system.
  2. The system records the information.
  3. The system prints a contract that the customer signs.

Use Case Example – Part Two

Exceptions:

1a. Customer identification is not recognized.
Clerk repeats step 1.

1b. The customer violates the CD Rental Limit business rule.
The clerk notifies the customer of the violation.
The use case is abandoned.

Business Rule:

CD Rental Limit
A customer can rent only three CDs at any one time.

Alternatives

3a. The printer jams.
The clerk fills out the contract by hand.
The use case exits.

Acceptance Tests Example

- **Rent a CD**—This is the main course.
- **Bad Customer ID**—Enter the customer ID wrong.
- **CD Rental Limit**—A customer has three CDs and rents another one.
- **Printer Jam**—Simulate a printer jam (maybe out of paper).

Scenario

- Given
  - Set of conditions
- When
  - Something happens
- Then
  - Something results

- Can create scenarios from use cases

Given / When / Then Example

- 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

Customer Data	
Name	**ID**
James	007

and CD has ID

and CD is not currently rented

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

Full Example (2)

- When a clerk checks out a CD:

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

Full Example (3)

- Then the CD is recorded as rented

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 (3)

- Given

Rental Fee Business Rule
Fee
$3

Rental Time Business Rule
Time
2 days

- When a clerk checks out a CD on:

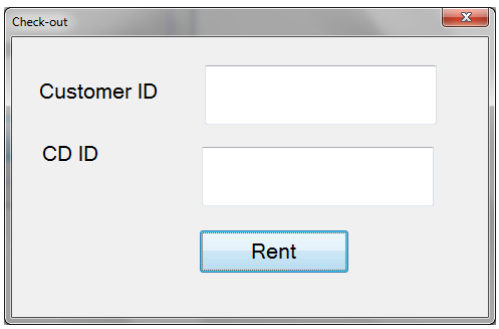
Today
1/1/2014

- 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);
```

Guidelines

Business Rule Tests

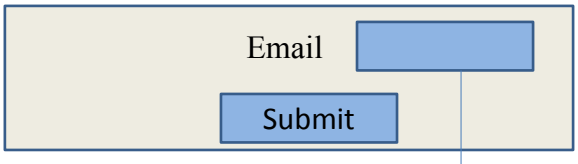
- Usually many business rules in use cases or scenarios
- Create separate tests for each business rule
  - Do not use flow tests
- Note:
  - If you find yourself wanting to "cut and paste" a flow test
    - Then there is probably a business rule underneath

Example of Business Rule Test

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

Rates		
Type	**Days**	**Cost?**
Regular	2	$2
Golden Oldie	3	$1.50
Hot Stuff	6	$12
Hot Stuff	50	IGBTYOT

User Interface and Business Rules



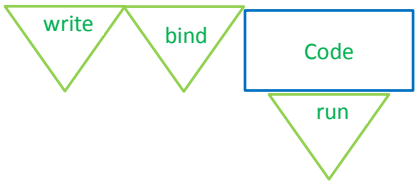
Middle-tier test

Call to middle-tier

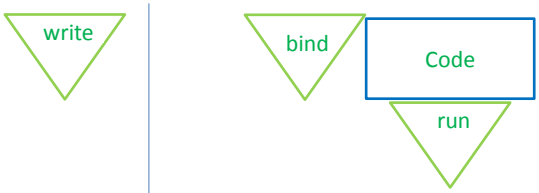
Email Domain Term with Middle-Tier validation	
Value	**Valid?**
manager@sam.com	yes
someone@@somewhere.com	no
!#$%&*+-/=?^_ {|}~@sam.com	yes
fred+filter@somewhere.com	yes

Process Note

Write within iteration



Write previous iteration



Not an Ending, But a Beginning

Recap

- Primary goals
  - Discover ambiguous and missing requirements
  - Create record of business/development understanding
  - Give feedback on quality
- Secondary goals
  - Executable regression test
  - Measure complexity of requirements
  - Use tests as basis for documentation
  - Measure progress toward “done”

Go Forth and Become Acceptance Test Creators

Thank you

Fit

Table Types

- Calculation Table
- Action Table
- Data Table

Calculation Table

Title			
Input Name 1	**Input Name 2**	**Result Name?**	**Notes**
Value for input 1	Value for input 2	Expected output	Anything that describes scenario
Another value for input 1	Another value for input 2	Another expected output	

Calculation Table Example

Discount Calculation		
Item Total	**Customer Rating**	**Discount Percentage?**
$10.00	Good	0%

Given

When

Then

Data Table

- 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

Action Table

- *Enter* enters data into an entry field
- *Press* initiates a process, such as a Submit button on a dialog box
- *Check* sees if a result is equal to an expected value

- Roughly equivalent to method call with parameters and return values

Action Name		
Enter	**Value Name 1**	Value for 1
Enter	**Value Name 2**	Value for 2
Press	**Submit**	
Check	**Value Name 3**	Expected value for 3

Action Table Example

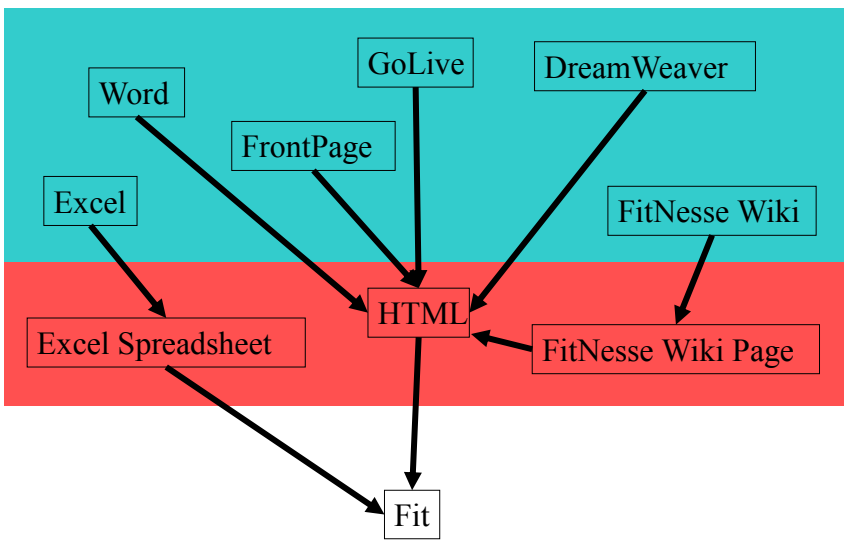
Can be just "When" or a stand-alone Given-When-Then

	Check Out CD		
Given	**Enter**	**Customer ID**	007
	Enter	**CD ID**	CD2
When	**Press**	**Rent**	
Then	**Check**	**Rented**	True

Fit Acceptance Test

- Fit (Framework for Integrated Testing)
  - Readable, writeable tests
  - Usually does not exercise system at UI layer
  - Can test at any layer
  - Fit is simple
    - Does very little
    - Can do so much with it

Tools for Writing a Fit Test



What Fit does to Your Tests

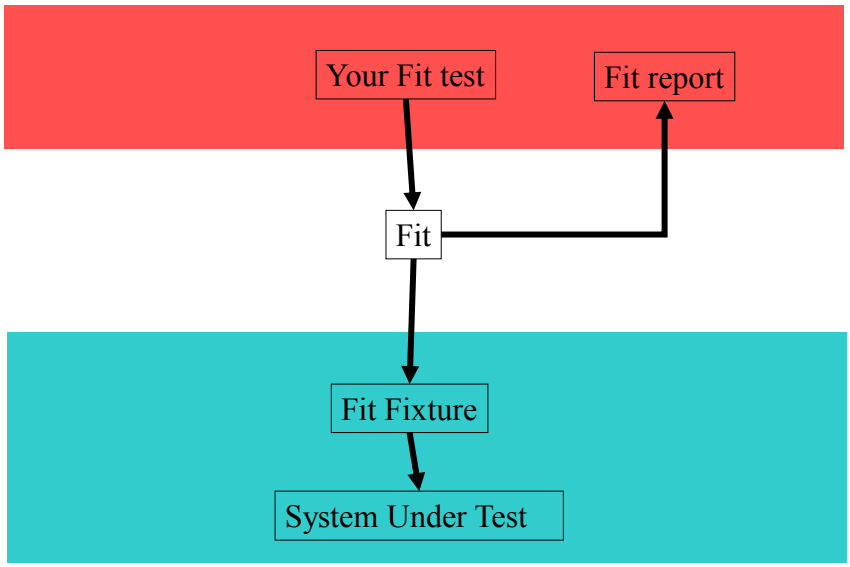


Table Equivalents

- Calculation Table = Column Fixture
  - Looks the same as previous examples
- Data Table = Row Fixture
  - Looks similar to previous examples
- Action Table = Action Fixture
  - Slightly different appearance

Fit Test 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%.

Discount		
Order total	Customer rating	Discount percentage?
10.00	Good	0
10.01	Good	1
50.01	Good	1
other	….	…..

Fit Code Example - Java

```
public class Discount extends fit.ColumnFixture
   {
   // Names must match column heads (sans spacing)
   public double orderTotal;
   public String customerRating;
   public int discountPercentage()
       {
       DiscountCalculator d = new DiscountCalculator():
       return d.computeDiscount(orderTotal,
          customerRating);
       }
   }

// customerRating could be type that is converted from a
   string,
```

Fit Code Example – C#

```
using fit;
namespace FitExampleDotNet {
 public class Discount : ColumnFixture  {
   public double OrderTotal;
   public String CustomerRating;
   public int DiscountPercentage() {
        DiscountCalculator d = new DiscountCalculator();
        return d.ComputeDiscount(
              OrderTotal, CustomerRating);
    }
  }
}
```


Fit Code Example - Java

```
public class Discount extends fit.ColumnFixture
   {
   // Names must match column heads (sans spacing)
   public Dollar orderTotal;
   public CustomerRating customerRating;
   public Percentage discountPercentage()
        {
        DiscountCalculator d = new DiscountCalculator():
        return d.computeDiscount(orderTotal,
           customerRating);
        }
   }

// Dollar, CustomerRating have from String methods (parse)
// Percentange must have to String and equals method
```

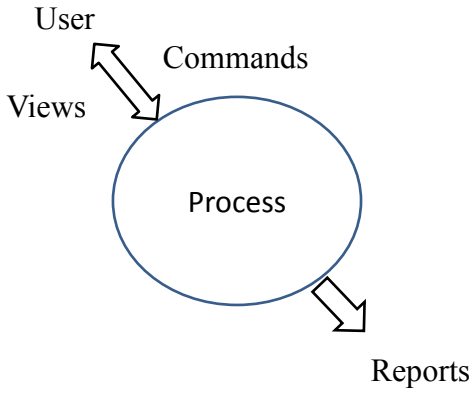
Fit Notes

- Values in tables could be file names
  - E.g. for complex data
- Fixture could execute a program, rather than call methods

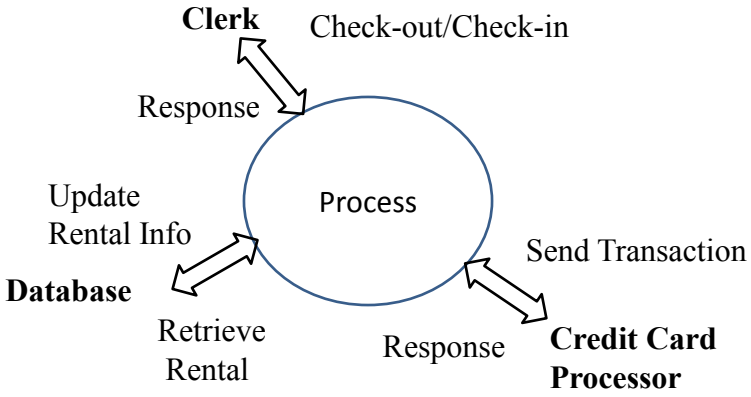
Context

Context Diagram

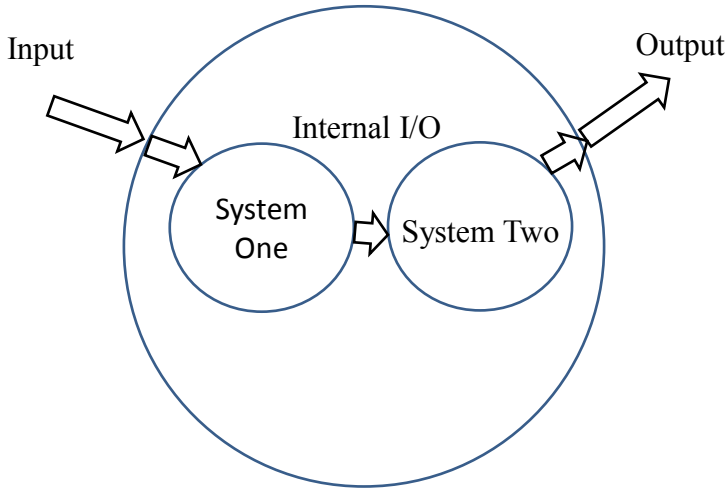
- A context diagram shows scope



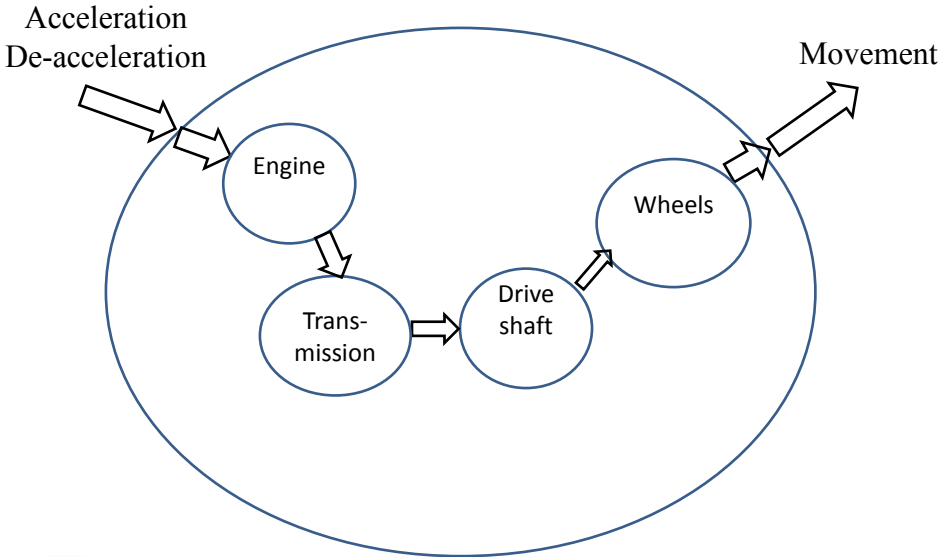
CD Rental Diagram



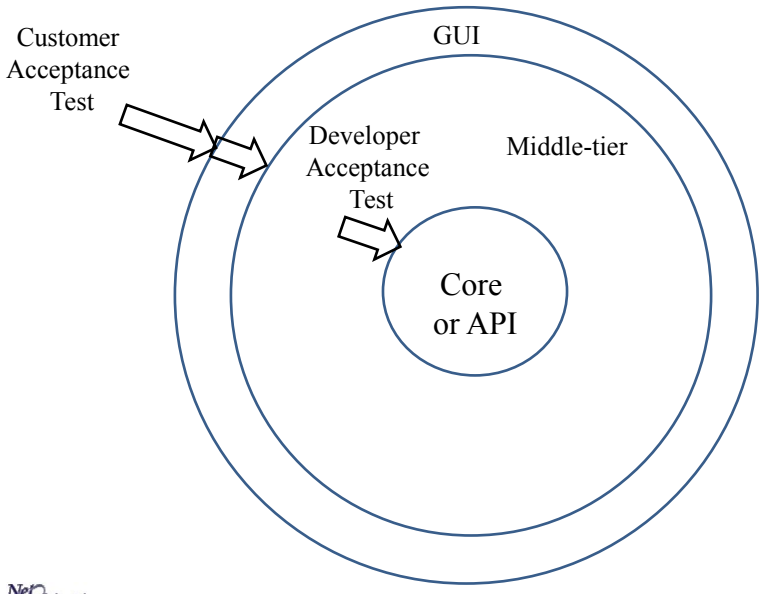
Embedded Contexts



Context of Automobile



Layered Context



75 Net Objectives 24 October 2014

Another ATTD Example

76 24 October 2014

First Example

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

Input and Output Example (continued)

Celsius	Fahrenheit	Notes
0	32	
100	212	Needed?
-273.15	-459.7	Precision?
-273.151	Error	Below 0 Kelvin
500	932	Maximum – Needed?

Input and Output – "Unit Tests"

Formula Tests

Celsius	Fahrenheit	Notes
0	32	
100	212	Needed?

Precision Tests

Celsius	Fahrenheit	Notes
-273.15	-459.67	Precision

Limit Tests

Celsius	Fahrenheit	Notes
-273.15	-459.67	0 Kelvin
-273.151	Error	Below 0 Kelvin
500	932	Maximum – Needed?