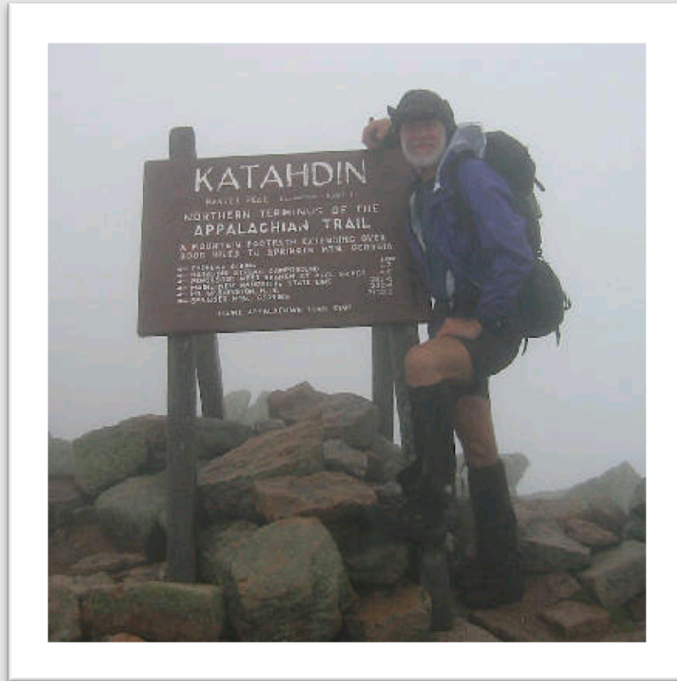


Accelerating DevOps with Behavior-Driven Development

Ken Pugh

Ken Pugh



BDD/ATDD, TDD, Design Patterns
SAFe Agile Software Engineering
Lean, Scrum, Kanban,
Training and Consulting



<http://atdd-bdd.com>



ken@kenpugh.com

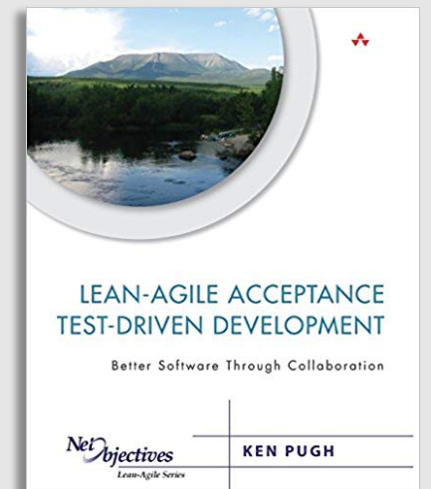


kenpugh



@kpugh

Lean Agile Acceptance Test-Driven Development: Better Software Through Collaboration



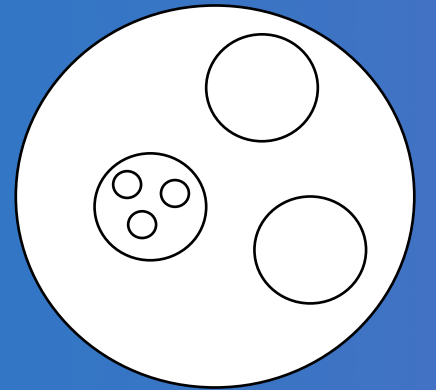
Overall Rule

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

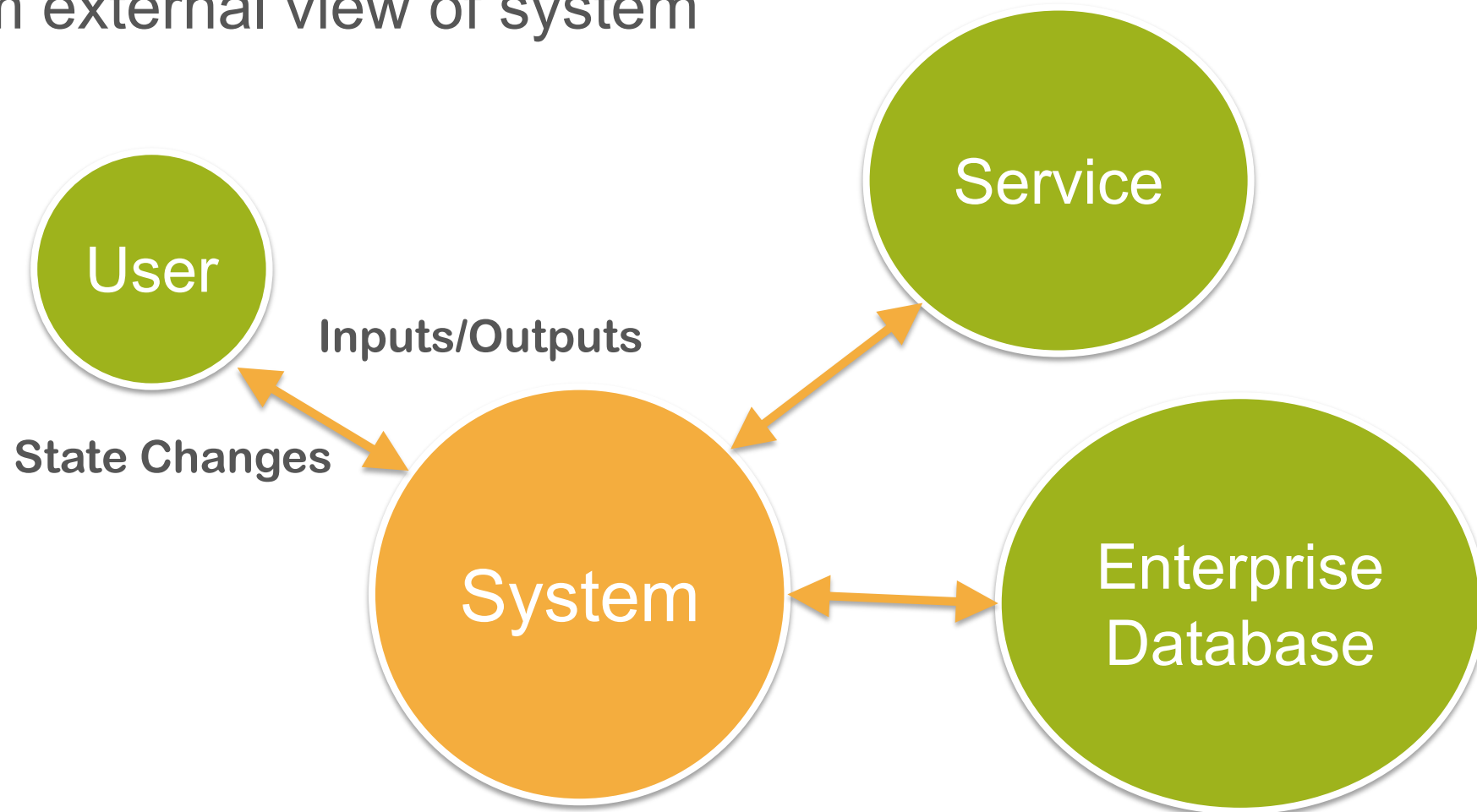
No code goes in till the test goes on



Introduction to Acceptance Tests/Behavior Driven Development

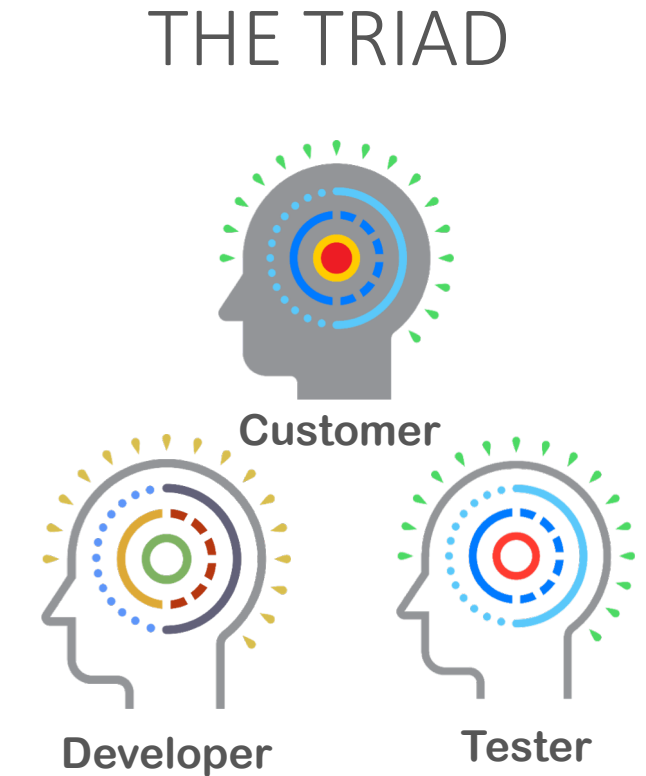
What Are Acceptance Tests?

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

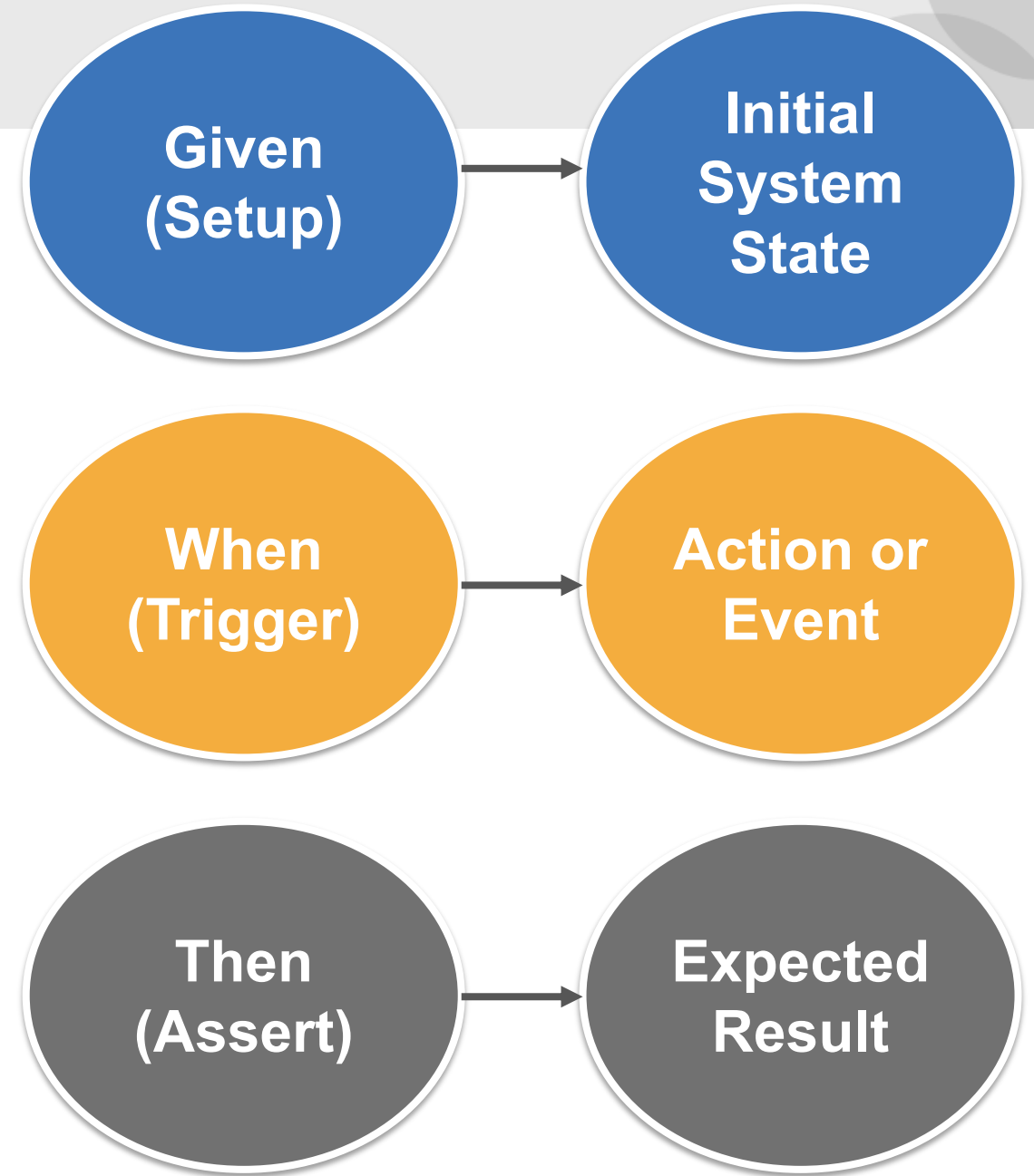
Who wants a fast car?

- ▶ 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



BDD Scenario Template

Given (Setup)

Car is not moving

When (Trigger)

Accelerator pressed

Then (Assert)

60 MPH reached before X seconds

Initial
System
State

Action
or Event

Actual
Result

Test

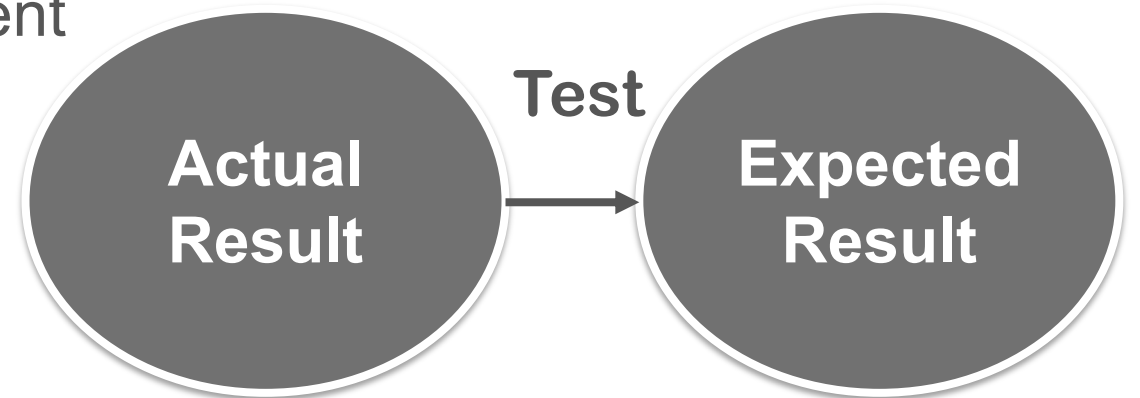
Expected
Result



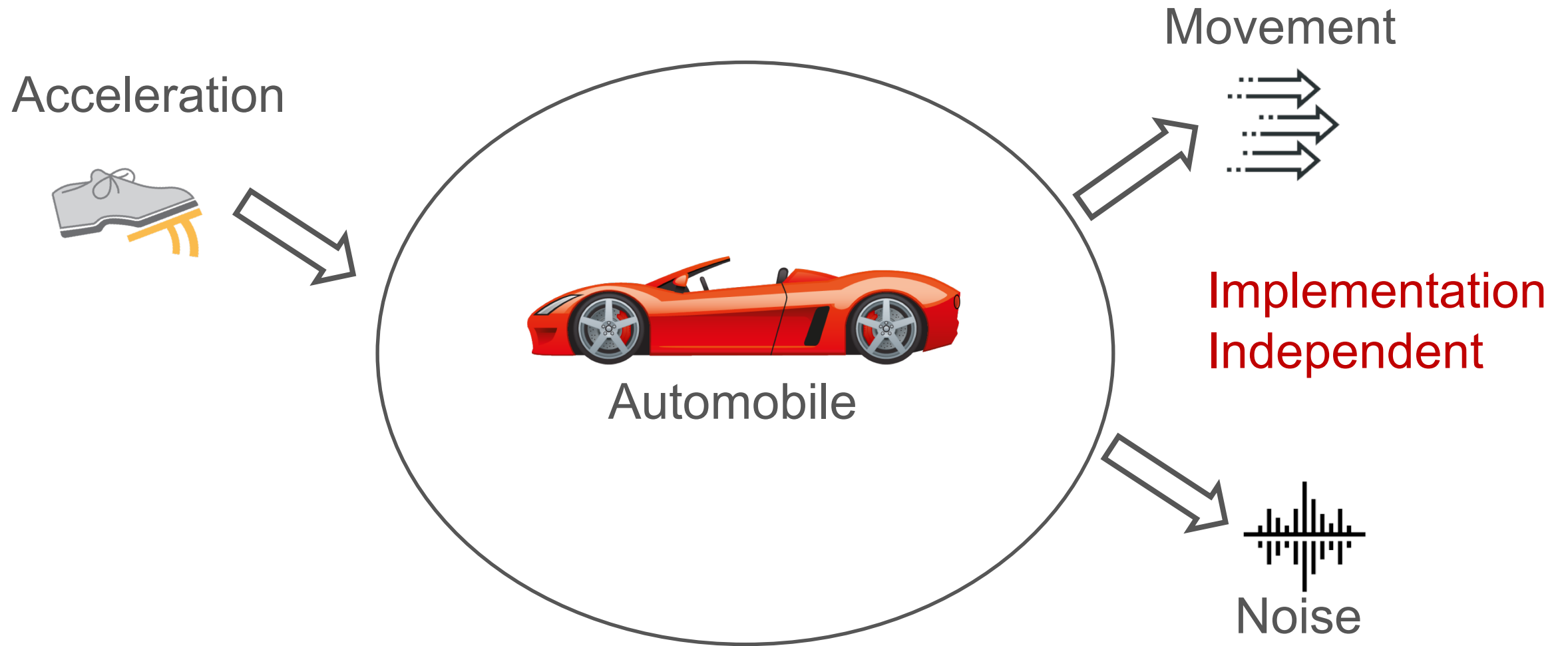
If “Then” is tested, scenario becomes an acceptance test

Term Alternatives

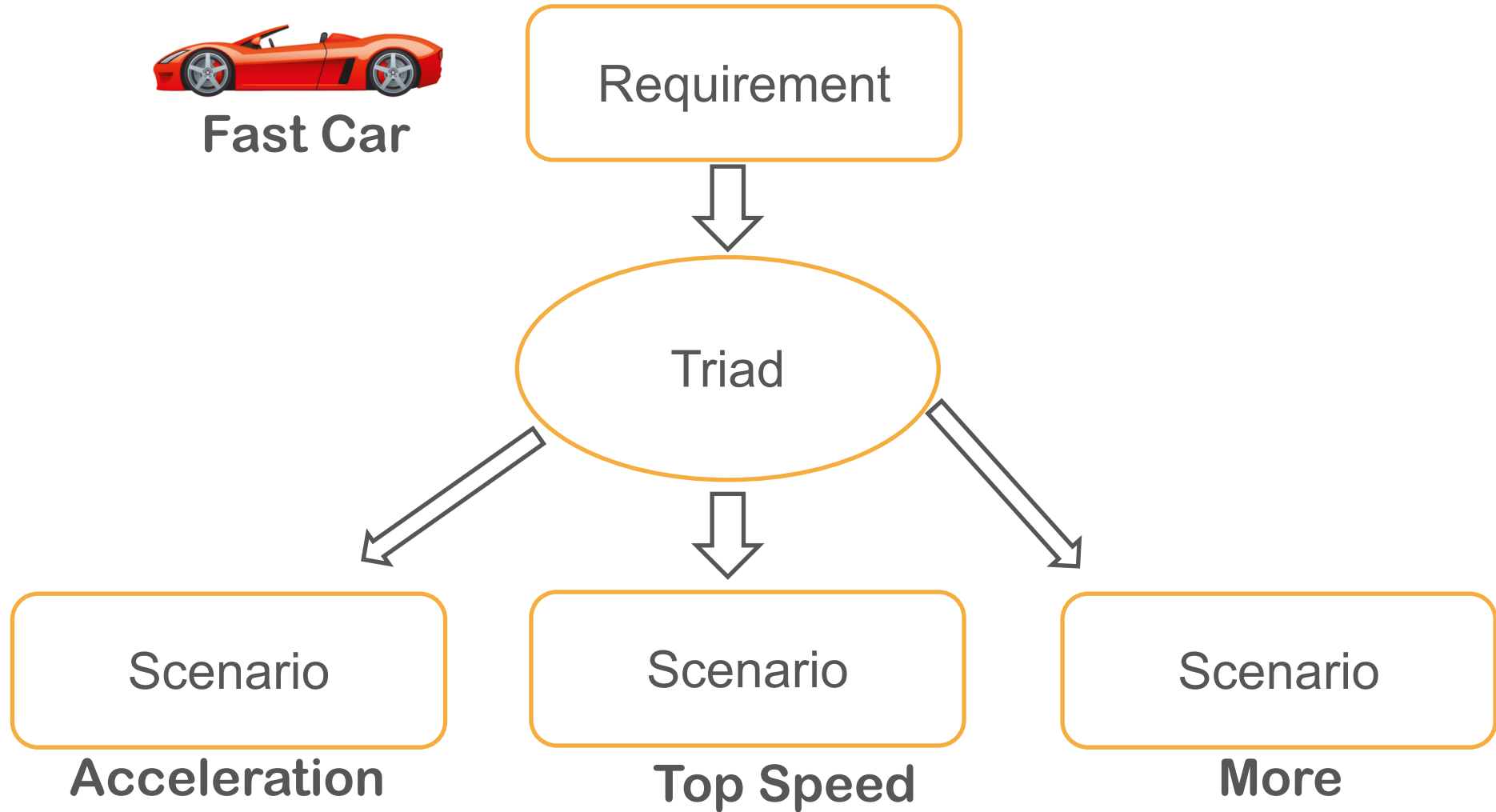
- ▶ Expected system state and output = behavior
 - Expected behavior drives development →
Behavior-Driven Development
- ▶ Tests that behavior is acceptable
 - Acceptance tests drive development →
Acceptance Test-Driven Development



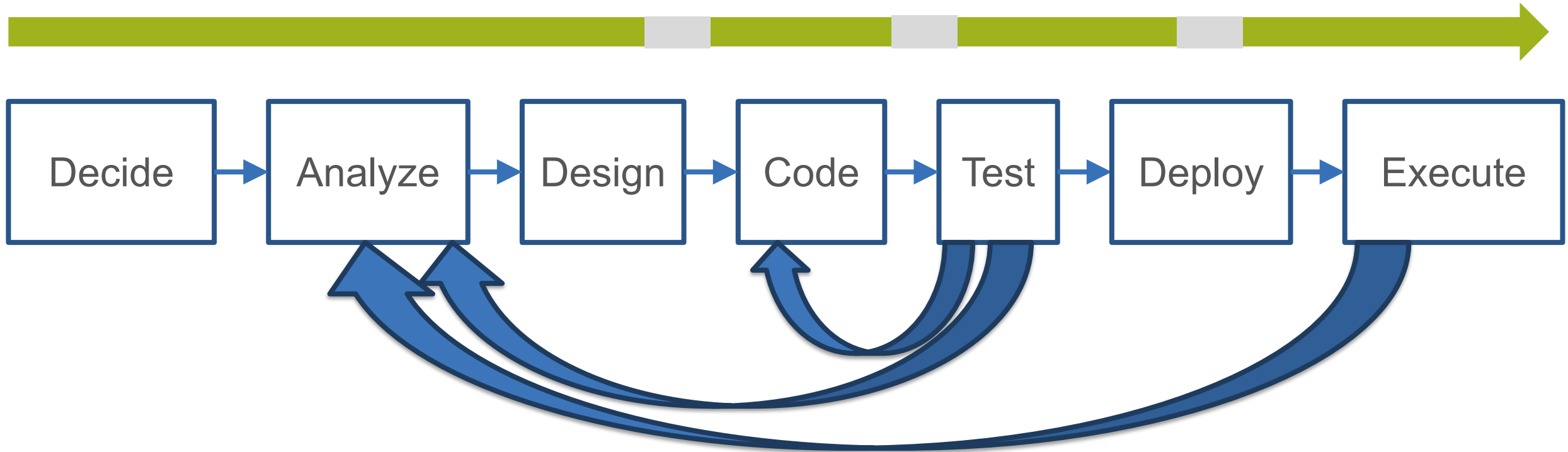
Example



BDD Discovery



DevOps Without BDD / ATDD



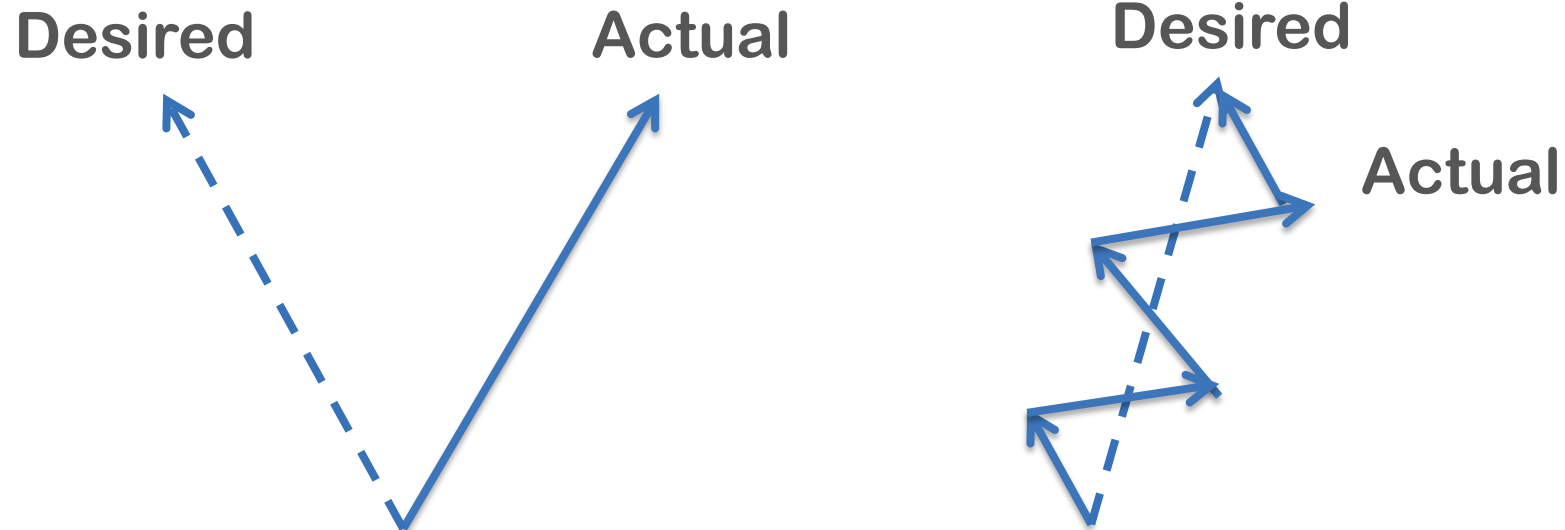
DEFECT: Not working right ☹️

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

Given

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%

When
Then

Ways To Implement Test

- ▶ Testing script
- ▶ Xunit framework
- ▶ BDD/ATDD framework

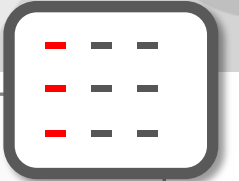
Creating the Script

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

Xunit Example

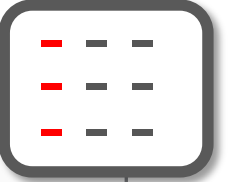


```
class TestCase {  
    void testDiscountPercentageForCustomer() {  
        SomeClass o = new SomeClass();  
        assertEquals(1, o.computeDiscount(50.01, Good));  
        assertEquals(0, o.computeDiscount(10.00, Good));  
        assertEquals(1, o.computeDiscount(10.01, Good));  
        assertEquals(1,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

BDD/ATDD Framework Example

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

ATDD/BDD Framework Example



Scenario Outline: Compute discount

Given total is <OrderTotal> and rating is <CustomerRating>

When discount computed

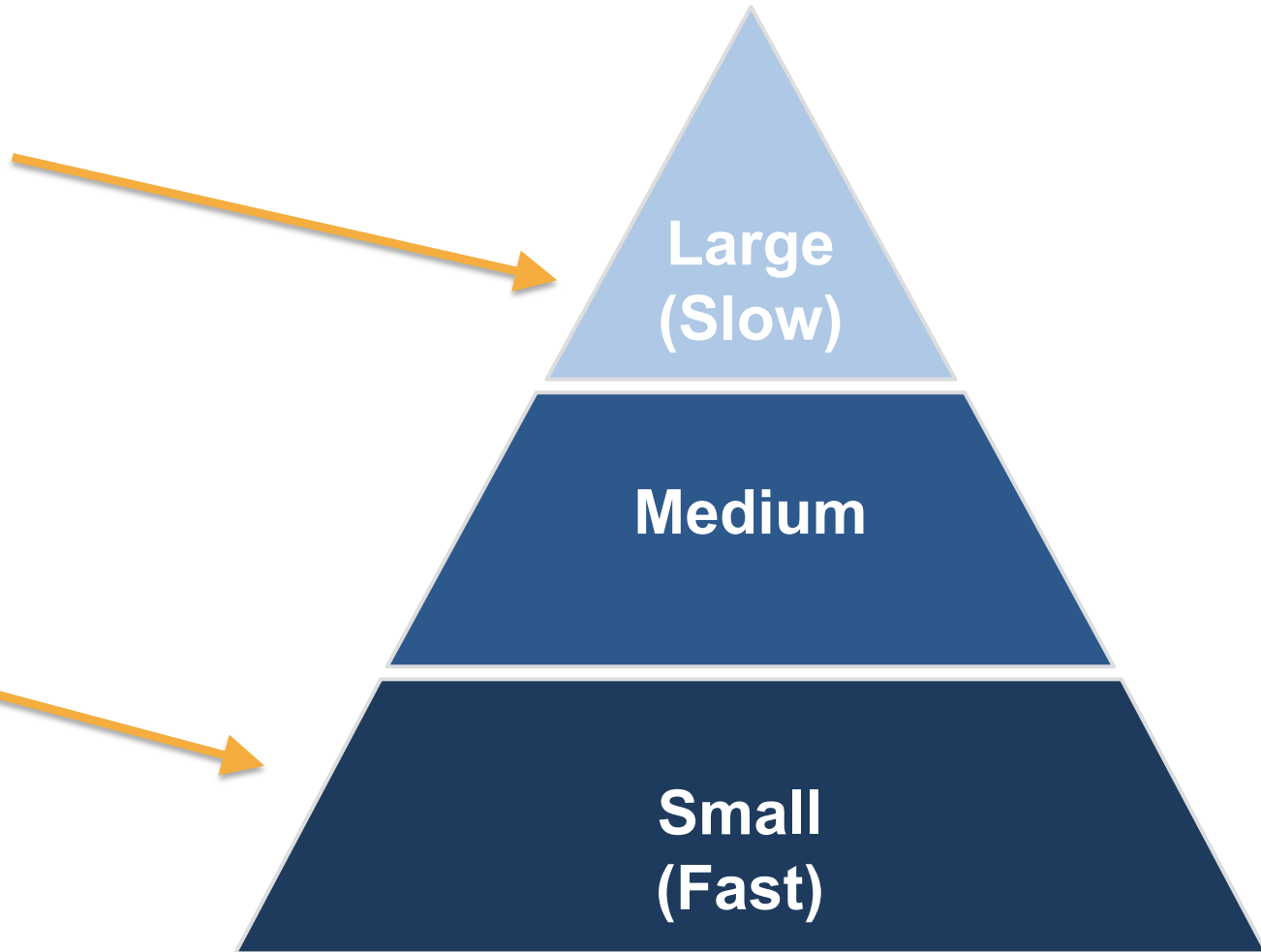
Then percent is <DiscountPercentage>

Examples:

OrderTotal	CustomerRating	DiscountPercentage
50.01	Good	1
10.00	Good	0
10.01	Good	1
0.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

Testing Pyramid

Testing script (1)



BDD/ATDD (6)

Domain Terms

Customer rating
Good
Excellent
Anything else?



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)

**Given
(Setup)**

**When
(Trigger)**

**Then
(Expected)**

Flow Test 1

Check Out Album

- ▶ Given Customer has ID

Customer Data	
Name	ID
James	007

and Album has ID

and Album is not currently rented

Album Data		
ID	Title	Rented
A2	Beatles Greatest Hits	No

Flow Test 2

- ▶ When a clerk checks out an Album:

Check Out Album		
Enter	Customer ID	007
Enter	Album ID	A2
Execute	CheckOut	

Flow Test 3

- ▶ Then the Album is recorded as rented

Album Data			
ID	Title	Rented	Customer ID
A2	Beatles Greatest Hits	Yes	007

and a rental contract is printed:

Rental Contract			
Customer ID	Customer Name	Album ID	Album Title
007	James	A2	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 an Album on:

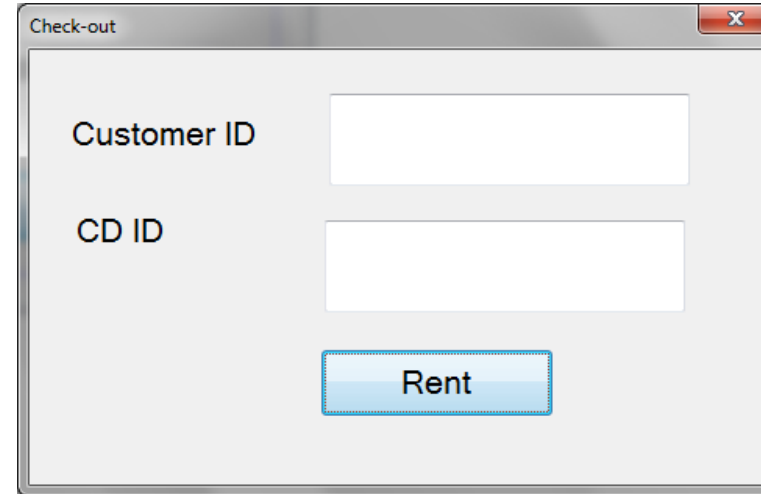
Today
6/1/2018

► Then a rental contract is printed:

Rental Contract					
Customer ID	Customer Name	Album ID	Album Title	Due	Fee
007	James	A2	Beatles Greatest Hits	6/3/2018	\$3

The Action

Check Out Album		
Enter	Customer ID	007
Enter	Album ID	A2
Execute	CheckOut	



- ▶ Can drive a GUI
- ▶ Or a method

```
CheckOut (CustomerID aCustomer, AlbumID anAlbum) ;
```

- ▶ 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

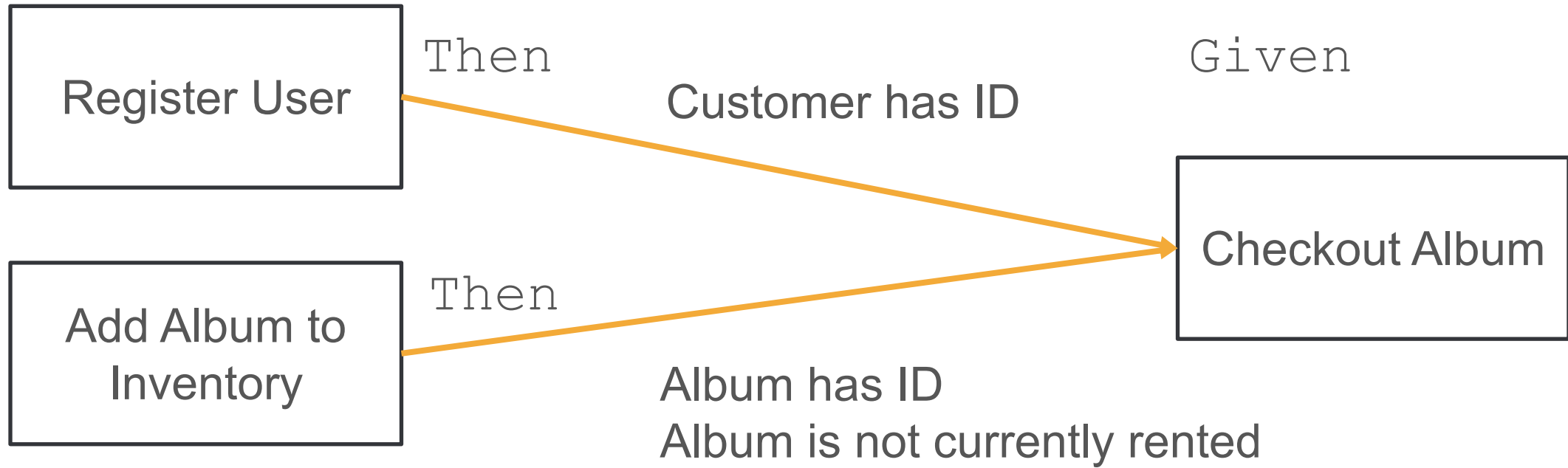
**Given
(Setup)**

**Given
Business
Rules**

**When
(Trigger)**

**Then
(Expected)**

Givens Are Somebody's Then





Requirements and Tests

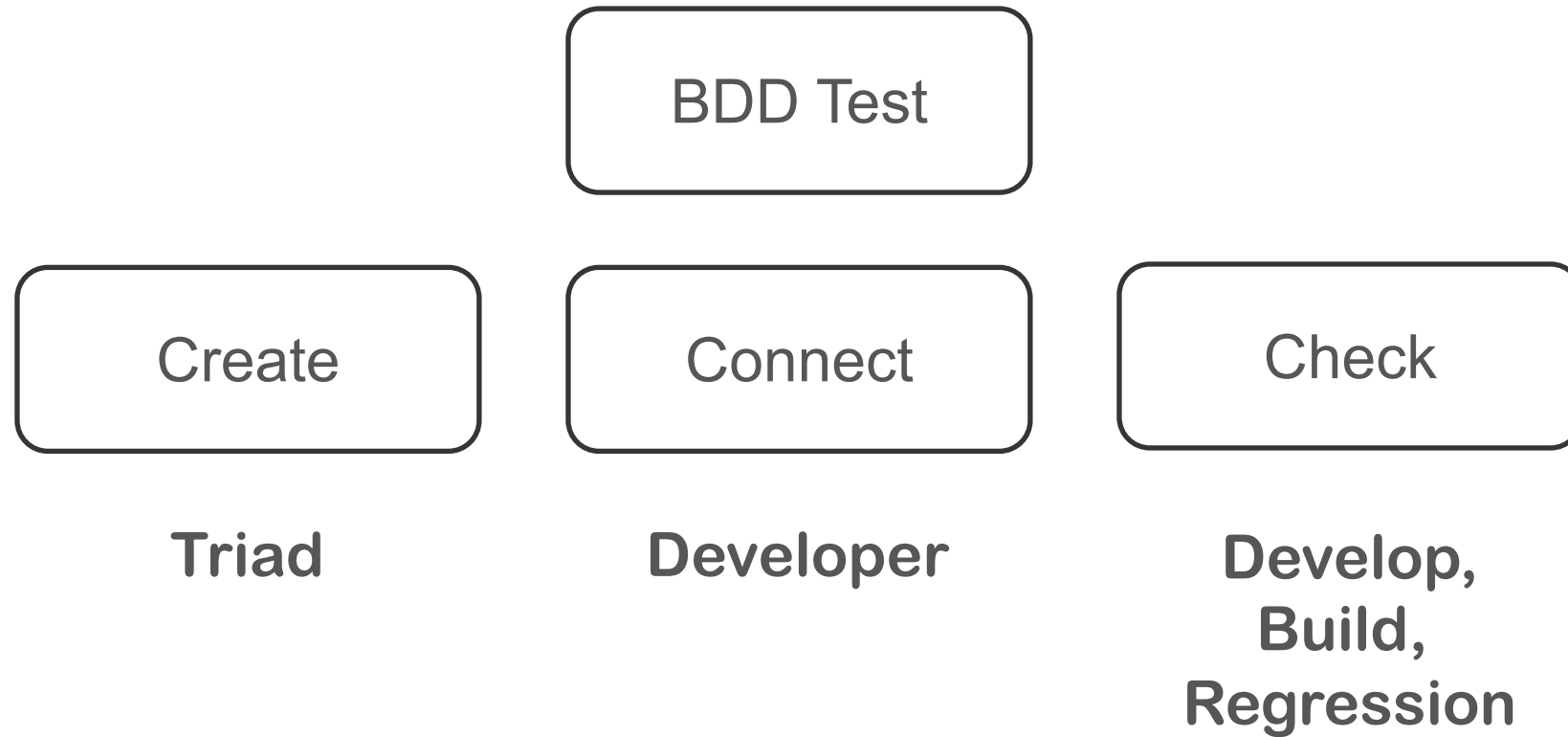
Types of Testing

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

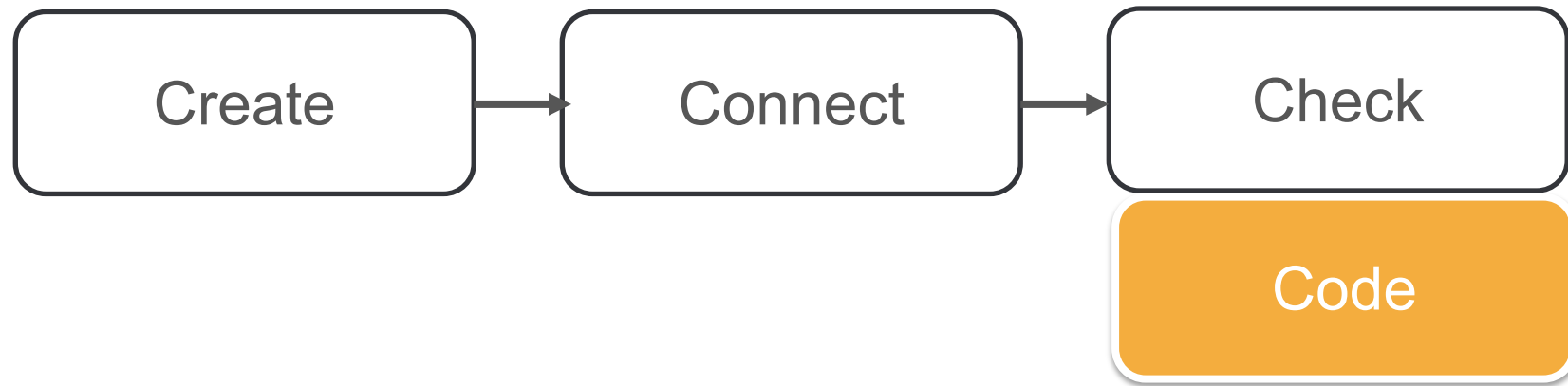
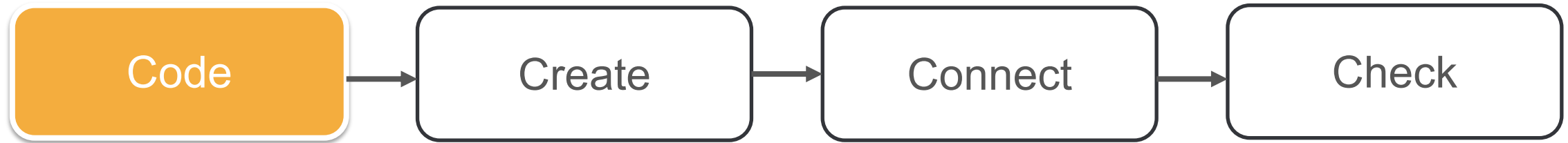
Requirements and Tests

- ▶ 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

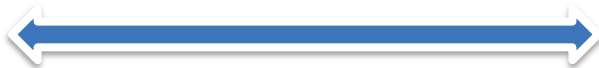

Requirements and Tests



Requirements and Test

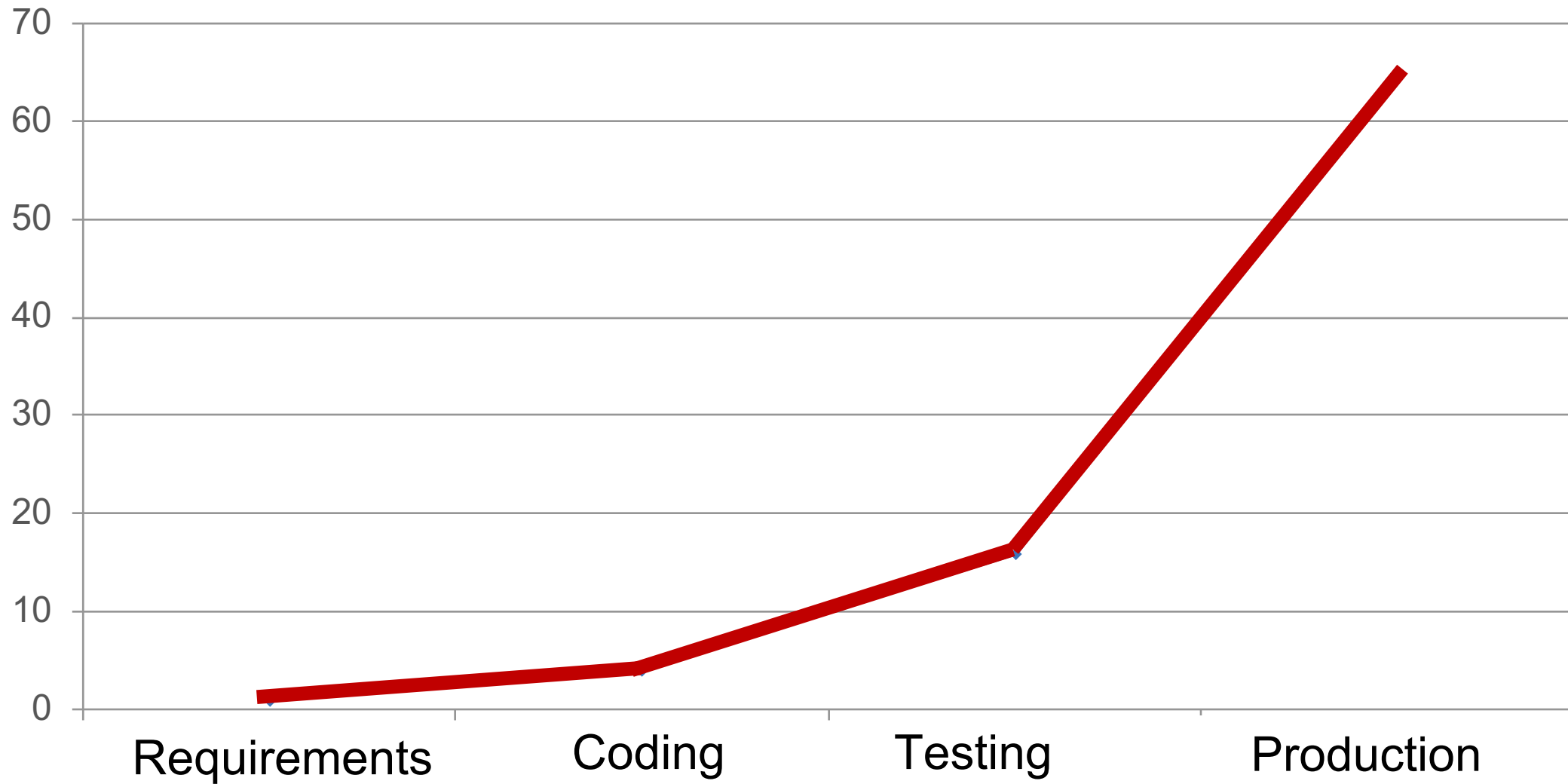


Tests

Kind of Behavior				
	Per Functionality			Cross Functional
<div>Business Facing</div>  <div>Technology Facing</div>	Customer/User Tests Business Intent	Pre-specified Usability e.g. corporate standard	Usability Is it pleasurable?	
	Component Tests Architectural Intent	Pre-specified Exploratory	Exploratory Is it self-consistent?	
	Unit Tests Developer Intent	Unit Property Testing	Property Testing Is it scalable, secure, responsive?	

Adapted from Mary Poppendieck, Brian Merrick, and Gerard Meszaros

Cost of Requirement Issue



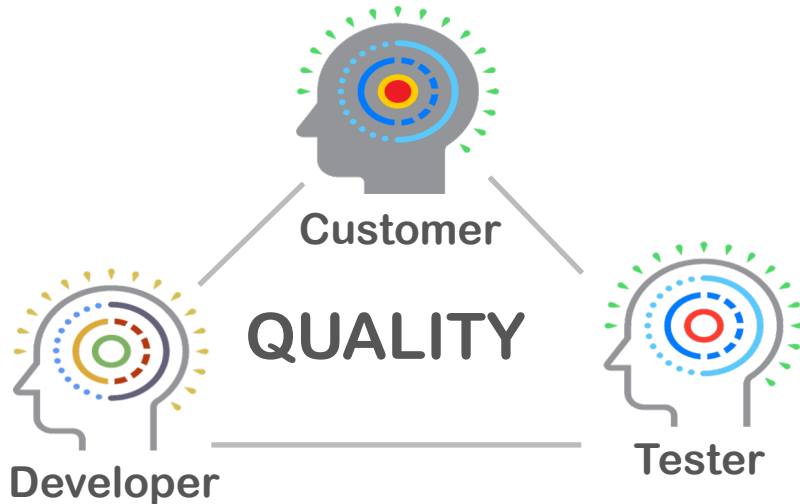
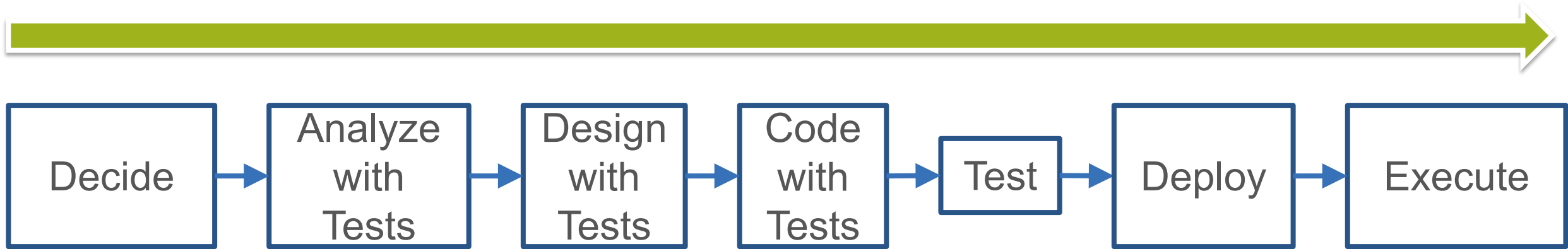
Could be 1 to 64, 1 to 256, or something else

Not an Ending, But a Beginning

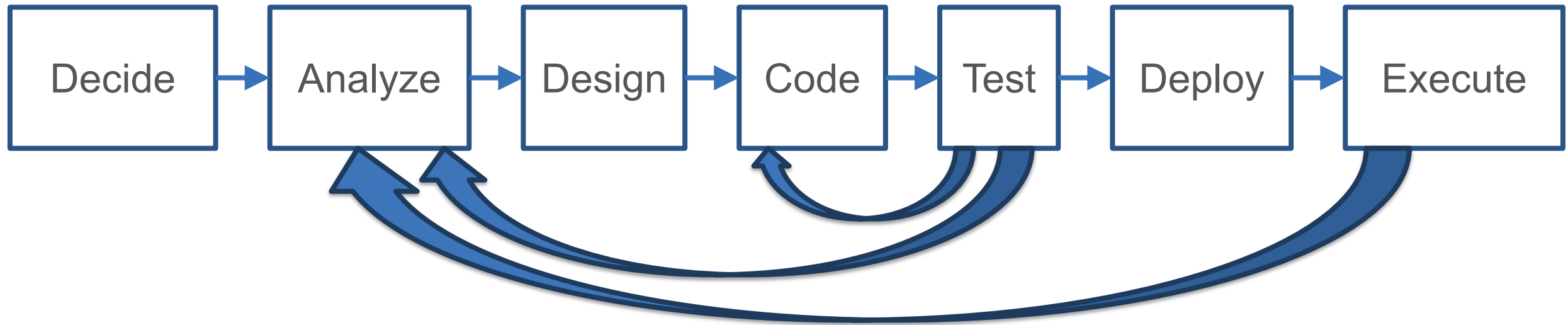


With BDD/ATDD

DevOps with BDD/ATDD



Replace Misunderstanding
with
Shared Understanding



► Right Tests - Reduce Loopbacks

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

► Automate Right – Eliminate Delays

Recap

▶ Primary goals





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

▶ Secondary goals

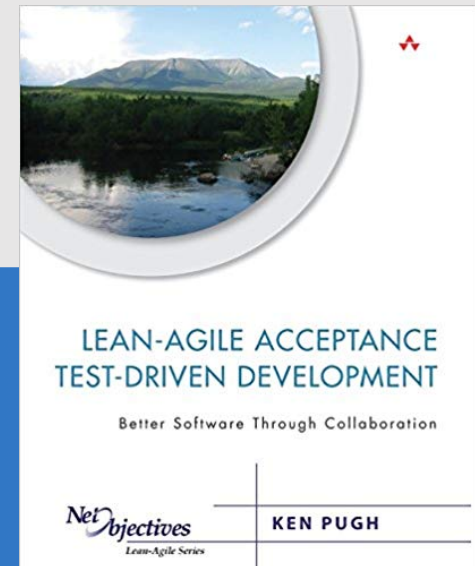
- Measure the complexity of requirements
- Use the tests as basis for documentation

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

 <http://atdd-bdd.com>
 ken@kenpugh.com
 [kenpugh](#)
 [@kpugh](#)

Go Forth and Become Behavior/ Acceptance Test Creators



Supplemental

Temperature Example

Try It Out

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

Celsius	Fahrenheit?	Notes
0	???	Freezing

Formula Tests

Celsius	Fahrenheit	Notes
0	32	
100	212	How many needed?

Precision Tests

Celsius	Fahrenheit	Notes
-273.15	-459.67	Precision

Limit Tests

Celsius	Fahrenheit	Notes
-273.15	-459.67	0 Kelvin
-273.16	Error	Below 0 Kelvin
500	932	Maximum – Needed?