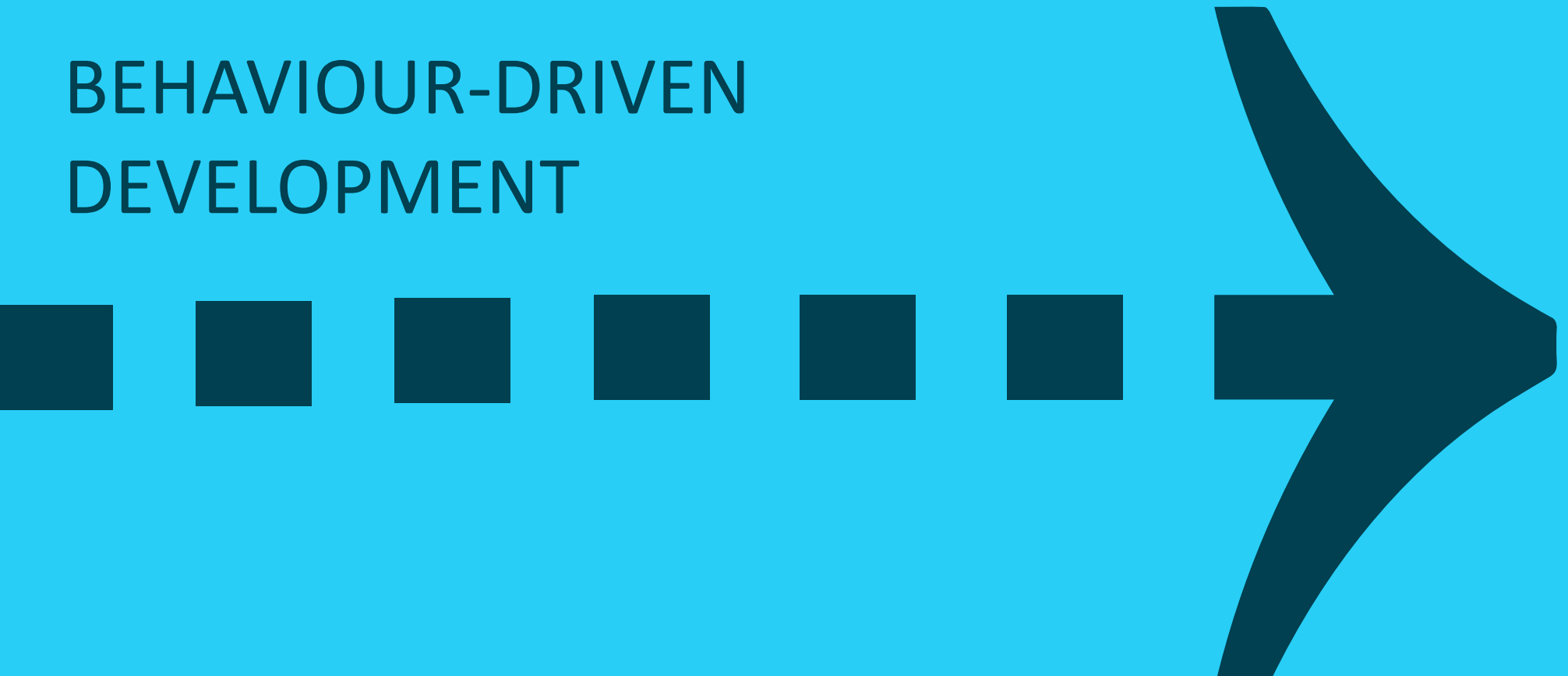
BEHAVIOUR-DRIVEN DEVELOPMENT

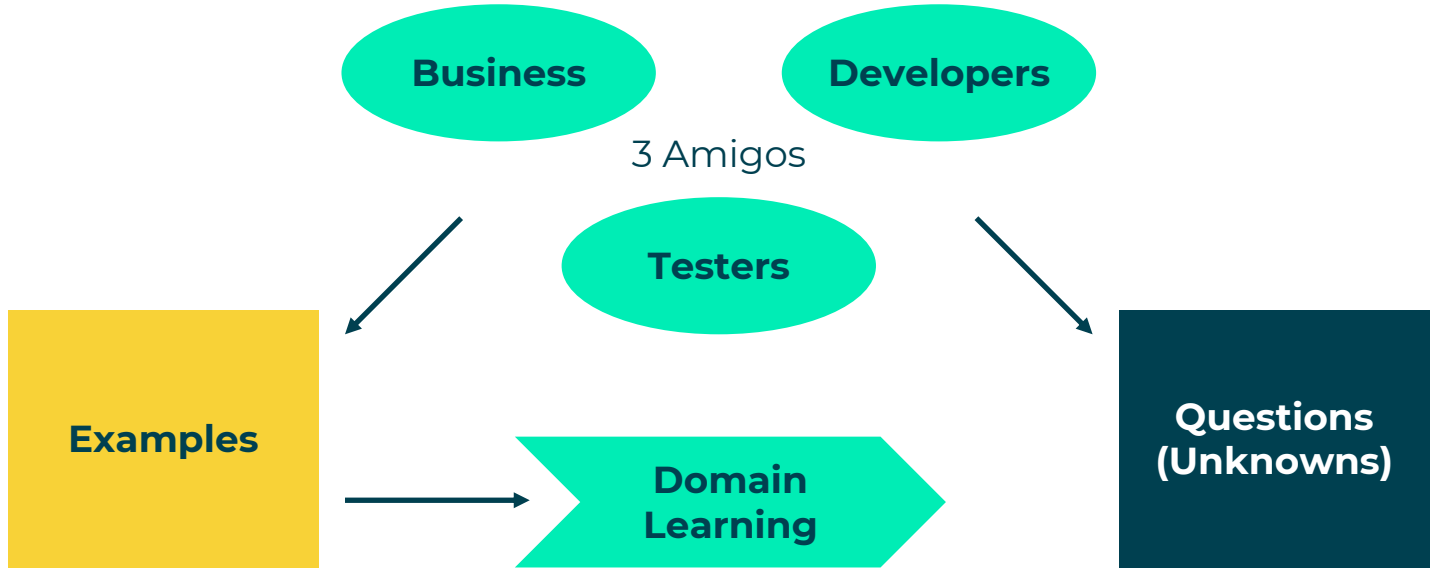


WHAT IS BDD?

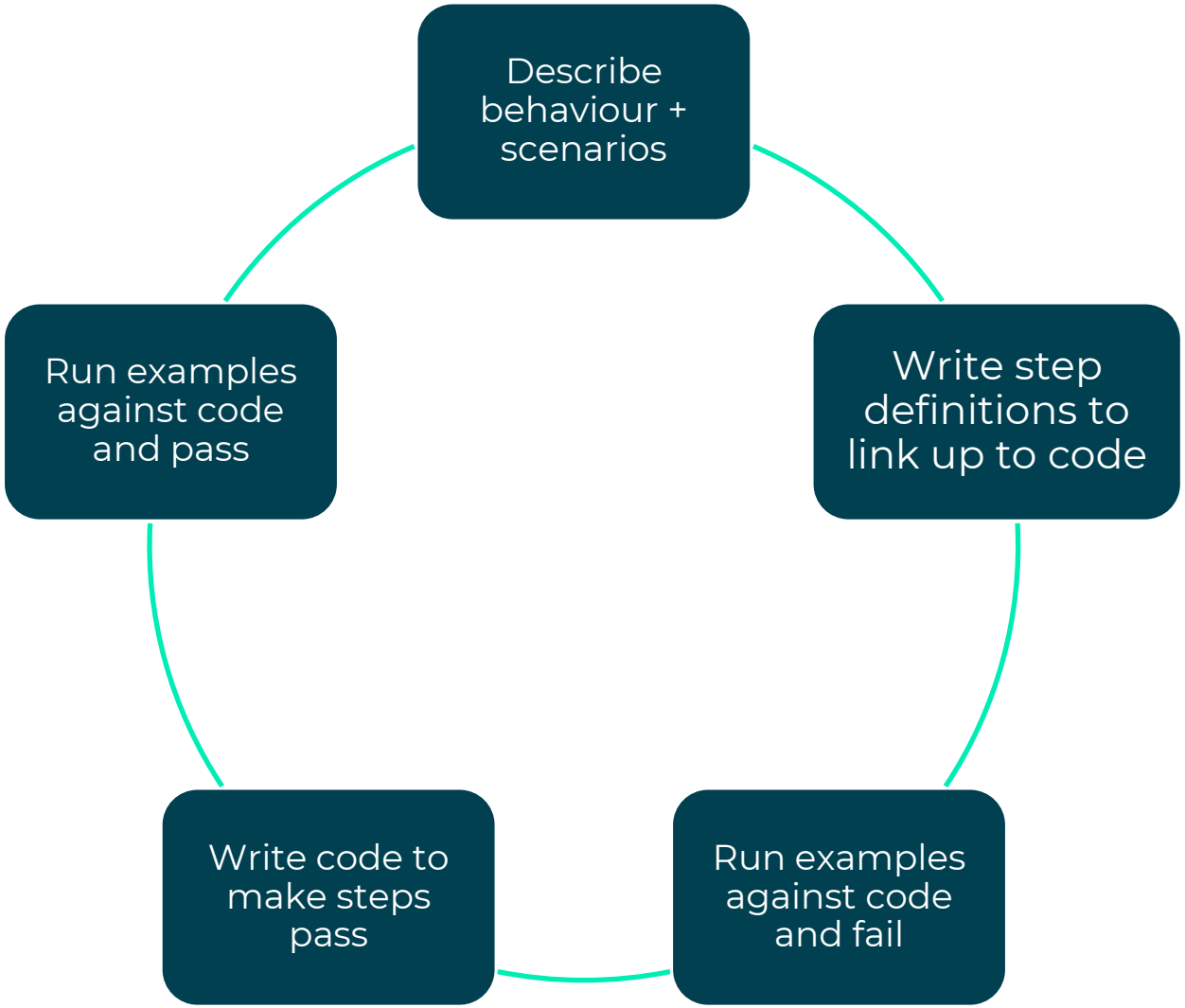
BDD is about conversation and collaboration

Language understandable to all stakeholders:

- Requirements = features
- Acceptance criteria = scenarios
- Scenarios illustrate how features work



BDD PROCESS


Describe behaviour + scenarios
Write step definitions to link up to code
Run examples against code and fail
Write code to make steps pass
Run examples against code and pass

WRITING SCENARIOS

1. **Write the ‘happy path’ scenario:**

   Given my shopping basket is empty

   When I add the book "BDD is Fun" to the basket

   Then the shopping basket contains 1 copy of "BDD is Fun"

2. **Then add alternative ‘edge’ scenarios:**

   Given my shopping basket contains the book "BDD is Fun"

   When I add another copy of book "BDD is Fun" to the basket

   Then the shopping basket contains 2 copies of "BDD is Fun"

BEST PRACTICES

- Write in present tense
- Always use business language
- Express the clauses as readable sentences
- Use only the situations shown in Figure 28 - ‘or’ doesn’t exist!
- Keep to one feature per story, and keep to max. 12 scenarios per feature
- Capitalise Gherkin keywords **Given, When, Then**, etc.
- Capitalise all titles

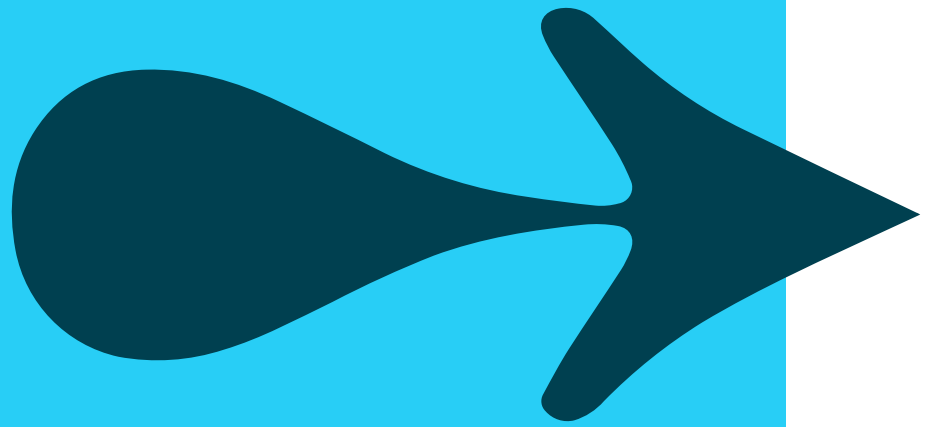
DEFINING SCENARIOS

Situation	Specific to feature	Description
Scenario	**<name>**	**Concise title**
Given	<assumption/context1>	Assuming a current state or context
And	<assumption/context2>	Additional context clauses
When	<event occurs>	When specific event(s) occur
And	<additional events occur>	
Then ensure this	<outcome occurs>	The following result(s)/outcome(s) should happen
And this	<additional outcome occurs>	

USING BACKGROUNDS

Use Background to avoid repeating the same clauses continually.

- Keep your Background section short and relevant
- Don’t use Background to set up **complicated states**, unless that state is actually something the client needs to know
- If the Background section has scrolled off the screen, think about using higher-level steps, or splitting the *.feature file
- Make your Background section vivid and try to tell a story



Scenario: User transfers cash from Savings to Checking
Given the user has entered a valid pin
And the account is open
Given I have 100 in checking
And I have 20 in savings

…

VALIDATING SCENARIOS

Ask these questions:

1. What is the intent?
2. Have we understood the expected behaviour?
3. Is the expected behaviour clearly expressed?
4. Can the rule be clearly understood when reading the scenario, and is there enough detail?
5. How many rules are defined in the scenario? Remember, it should ideally just be one!

WRITING USER STORIES

A user story usually contains:

- Definition
- Acceptance criteria

Definitions typically use the '**Connextra**' format:

As an X
I want Y
So that Z

Acceptance criteria are conditions that must be satisfied for the story to conclude successfully

- **Definition of done**
- **Makes the story testable**

USING EXAMPLES

User stories with raw acceptance criteria can miss the overall intent.

Examples of how it should work make things clearer:

- ‘In the best case scenario, this is how it should work…’
- ‘It could also work like this…’
- ‘Even this could happen! This is what the response should be…’

Examples illustrate:

- What could happen
- Under what circumstances it might happen
- What should be done when it happens

CUCUMBER AND GHERKIN

‘Cucumber is a tool that supports Behaviour-Driven Development (BDD).’

Cucumber reads executable specifications written in plain text and validates that the software does what those specifications say.

The specifications consist of **scenarios**.

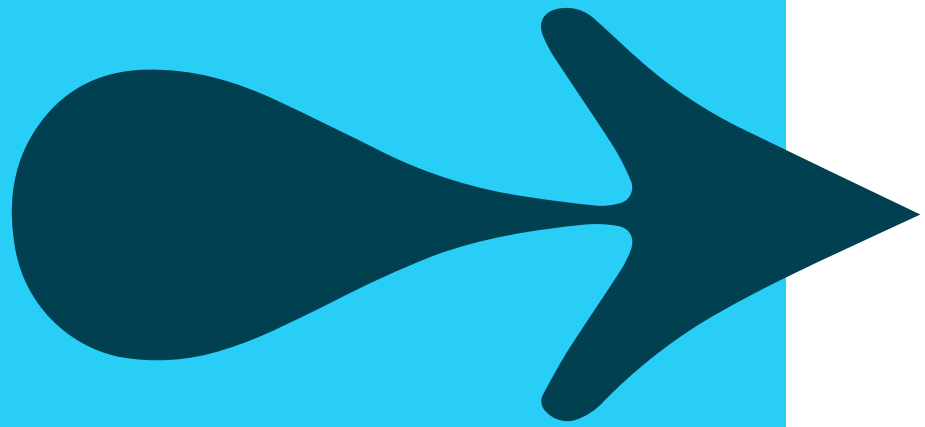
The scenarios must follow some basic syntax rules, called **Gherkin**.

STEP DEFINITIONS EXAMPLES

Scenarios are broken into step definitions:

Given the balance is **£500**
When the user withdraws **£200**
Then the balance is **£300**

APPLYING GHERKIN TO ACCEPTANCE CRITERIA



Customer requirement: ‘Books can be added to the cart’

Gherkin would describe the criteria more formally as:

Given my shopping basket is empty
When I add the book ‘BDD is Fun’ to the basket
Then the shopping basket should contain 1 copy of 'BDD is Fun'

- Feature
- Background
- Scenario
- Given
- When
- Then
- And
- But

Gherkin keywords

AUTOMATION

- Developers can use a **Cucumber** engine to run Gherkin. This engine may link to a framework to talk to browsers, simulators etc.
- Mobile developers use tools like **Appium**
- Web developers can use **Selenium** to hook into browsers. Step definitions run commands via Selenium API.

STEP DEFINITIONS

```
@Given("the balance is $bal")
public void setBalance(double bal)
{
    myCal = new Calculator(bal);
}

@When("the user withdraws $amount")
public void withdrawAmount(double amount)
{
    myCal.Withdraw(amount);
}

@Then("the balance is $bal")
public void testResult(double bal)
{
    Assert.assertEquals(bal,
    myCal.getBalance());
}
```

An example – Write the CUT

```
public class Calculator {

    public int firstNumber;
    public int secondNumber;

    public int Add() {
        return firstNumber + secondNumber;
    }

    public int Multiply() {
        return firstNumber * secondNumber;
    }
}
```

An example… Define features

```
Feature: Calculator
![Calculator] (https://specflow.org/wp-content/uploads/2020/09/calculator.png)
Simple calculator for adding** two**numbers

Link to a feature: [Calculator](SpecFlowProject4/Features/Calculator.feature)
* **Further read * **: **[Learn more about how to generate Living Documentation]
(https://docs.specflow.org/projects/specflow-livingdoc/en/latest/LivingDocGenerator/Generating-Documentation.html)**

    @mytag
    Scenario: Add two numbers
        Given the first number is 50
        And the second number is 70
        When the two numbers are added
        Then the result should be 120
```

An example... Define features

Scenario: Add Lots of numbers

Given the first number is <n1>
And the second number is <n2>
When the two numbers are added
Then the result should be <n3>

Examples:

n1	n2	n3
1	2	3
11	12	23
21	22	43
31	32	63

Parameterised tests

Scenario: Multiply two numbers

Given the first number is 5
And the second number is 6
When the two numbers are multiplied
Then the result should be 30

Add more scenarios

An example… write the steps

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TechTalk.SpecFlow;
[Binding]
public sealed class CalculatorStepDefinitions {
    private Calculator calculator;
    private readonly ScenarioContext _scenarioContext;

    public CalculatorStepDefinitions(ScenarioContext scenarioContext) {
        _scenarioContext = scenarioContext;
       calc = new Calculator();
    }

   [Given("the first number is (.*)")]
   public void GivenTheFirstNumberIs(int number)  {
        calculator.firstNumber = number;
  }

  [Given("the second number is (.*)")]
  public void GivenTheSecondNumberIs(int number)  {
        calculator.secondNumber = number;
 }
```

```
Can also use _scenarioContext to store values:
_scenarioContext["result"] = calculator.Add();
```

```
int result;
[When("the two numbers are added")]
public void WhenTheTwoNumbersAreAdded()  {
        result = calculator.Add();
}

[When("the two numbers are multiplied")]
public void WhenTheTwoNumbersAreMultiplied()  {
        result = calculator.Multiply();
}

[Then("the result should be (.*)")]
public void ThenTheResultShouldBe(int expectedResult)  {
        Assert.AreEqual(expectedResult, result);
}
```

LAB

"Behaviour Driven Development (BDD)" Lab

Afterwards, we'll discuss what you thought...