



# From Manual to Automating UI Testing

## Module 2: Web Front-end test automation

Franck Théolade - ADM

[frtheola@Microsoft.com](mailto:frtheola@Microsoft.com)



## Brief BDD overview

- Gherkin fundamentals
- Acceptance Tests
- Best Practices
- Exercise 1: Defining Bing Search Scenario

## Understanding Web UI Testing


- Anatomy of Web page / DOM elements
- Exercise 2: Finding elements with browser
- Designing Testable Web Pages
- Exercise 3: First Selenium Automated test

## SpecFlow Fundamentals

- Testing Stack
- Features, scenarios, Binding, step definitions & hooks
- Sharing data, steps and best practices
- Hands-on Labs: Bing Search Scenarios

## Next Steps

- Practice tips & References
- Feedback to next session
- Getting ready for next session



# Cucumber, Gherkin and Behaviour Driven Development



Microsoft Services



# What is Automation Testing?



Process in which software tools execute pre-scripted tests on a software application before it is released into production.



Automated testing is code-driven and mitigates business risks

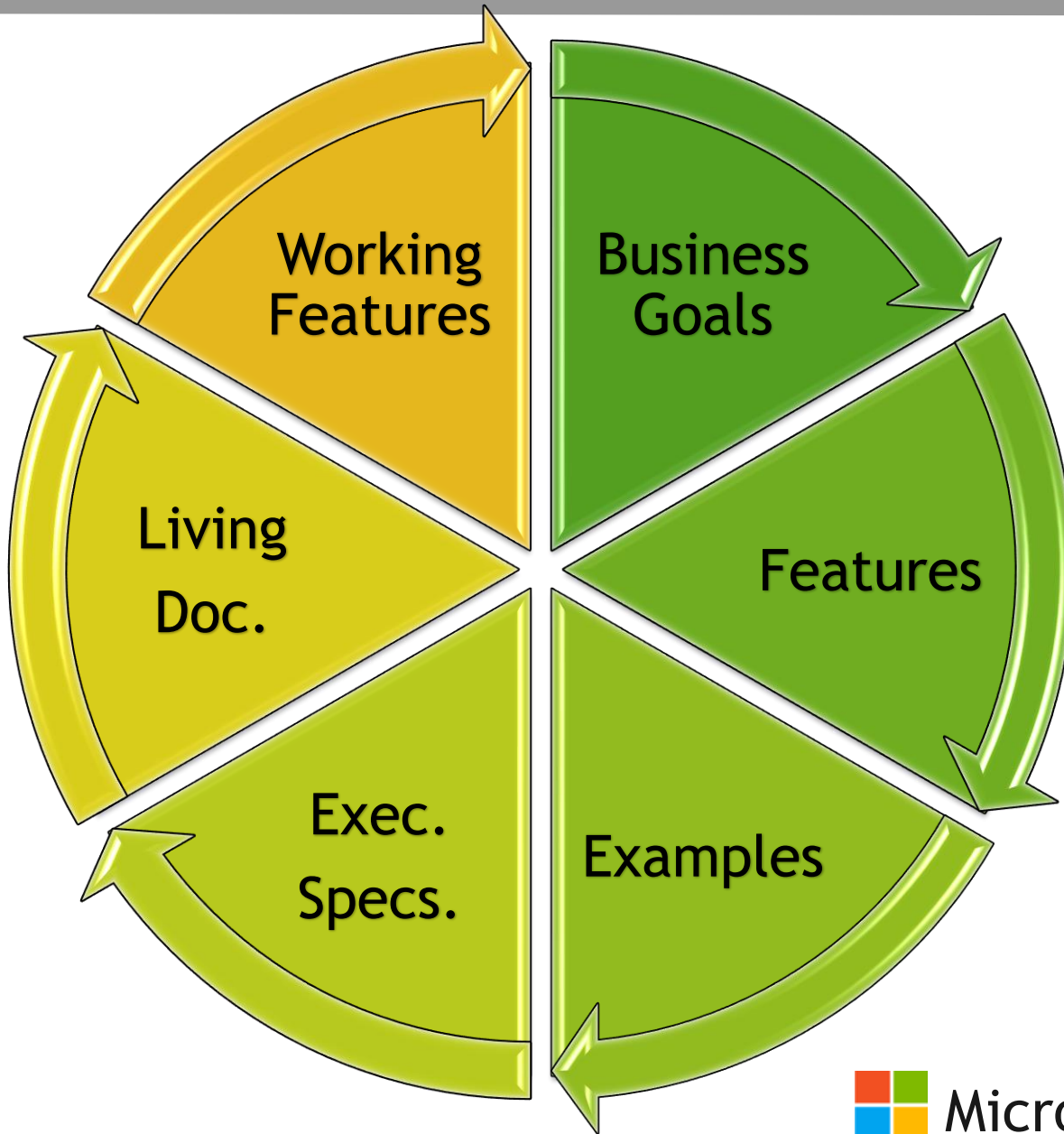


Simplify as much of the testing effort as possible with a minimum set of scripts.



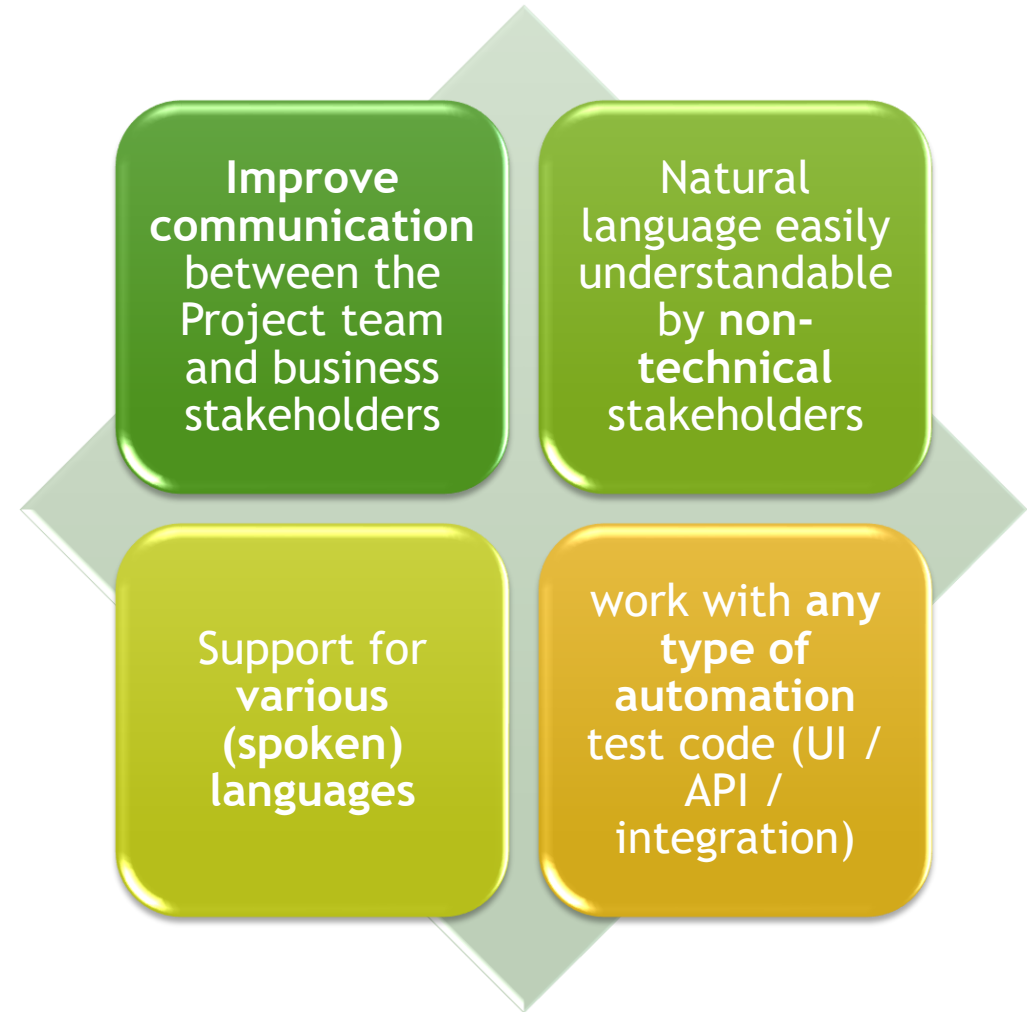
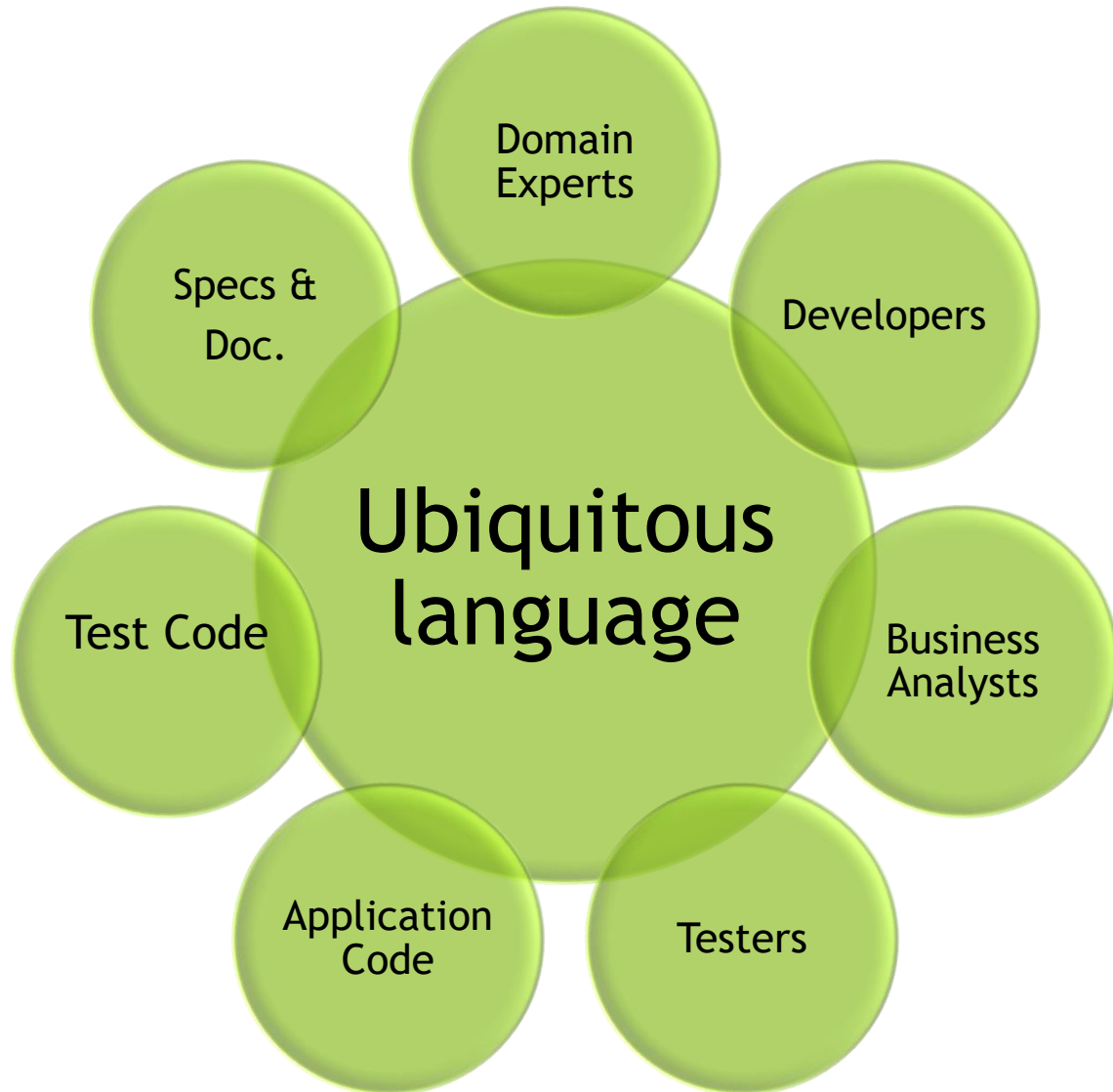
The method or process being used to implement automation is called a test automation framework.

# Behaviour Driven Development



<https://aka.ms/BDDWorkshop>

# BDD - From Ideas to working software



# Gherkin, Cucumber and SpecFlow

*“The hardest single part of building a software system is deciding precisely what to build”*



Gherkin is plain-text English with some structure.



Cucumber understands and parses the Gherkin Language.



Specflow is Cucumber for the .NET Framework.

# Exercise 1

## Specifying Bing Search Scenario with Gherkin

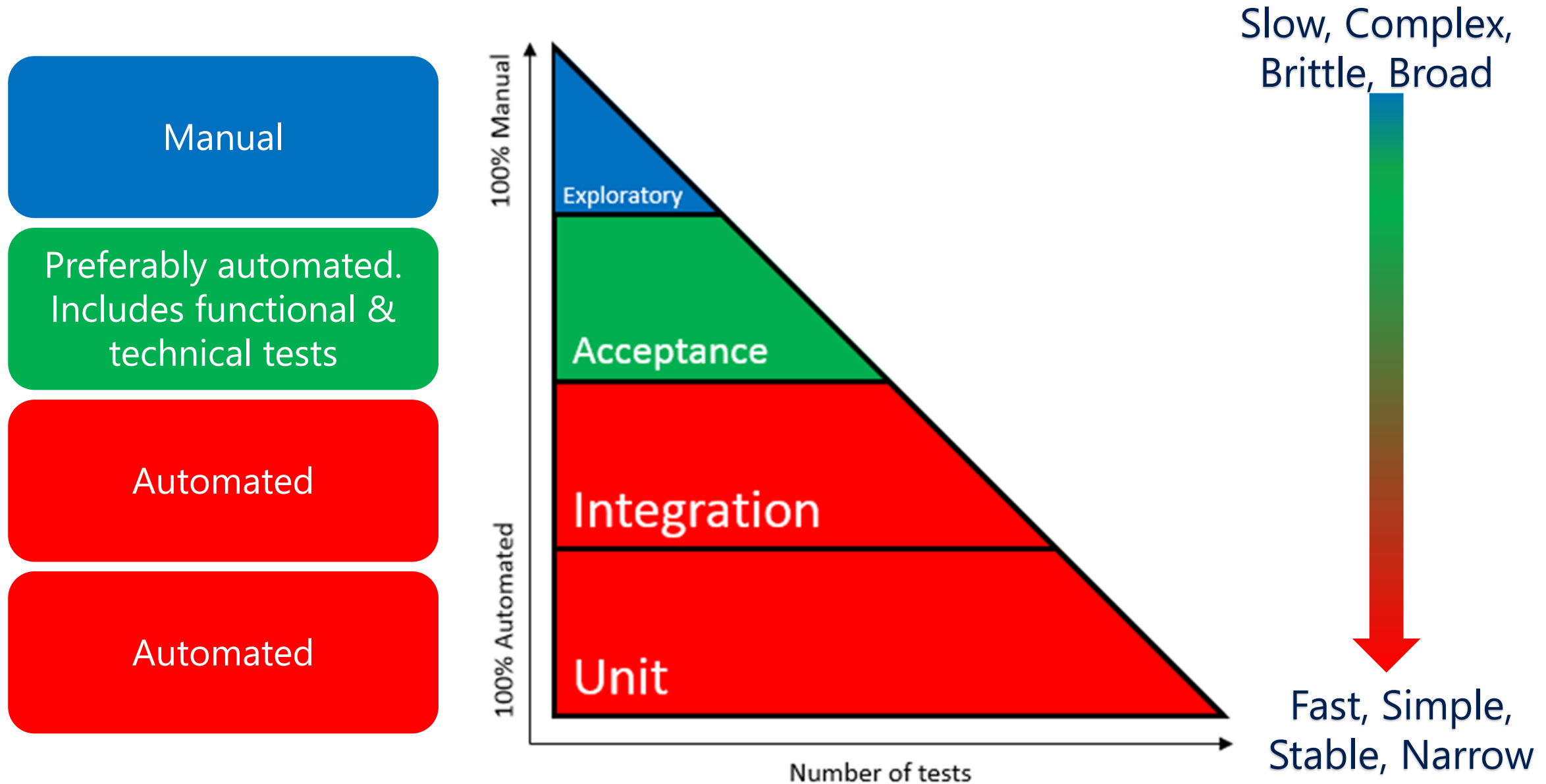
```
public void IsRecurring()  
{  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
    Assert.IsTrue(customer.IsRecurringSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargedResultToFailOnceShouldBeCurrent()  
{  
    //Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    customer.RecordChargedResult(false);  
  
    // Assert  
    Assert.IsTrue(customer.HasCurrentSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargeResultToFailMaximumTimesAndNoLongerCurrent()  
{  
    // Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    for (var i = 0; i < Customer.MAX_PAYMENT_FAILURES; i++)  
        customer.RecordChargedResult(false);  
  
    //Assert  
    Assert.IsFalse(customer.HasCurrentSubscription);  
}
```



# Testing Pyramid



# From Acceptance Testing to Living Documentation



Business/user/customer point of view

Validates that the right system is being built

Pass/fail



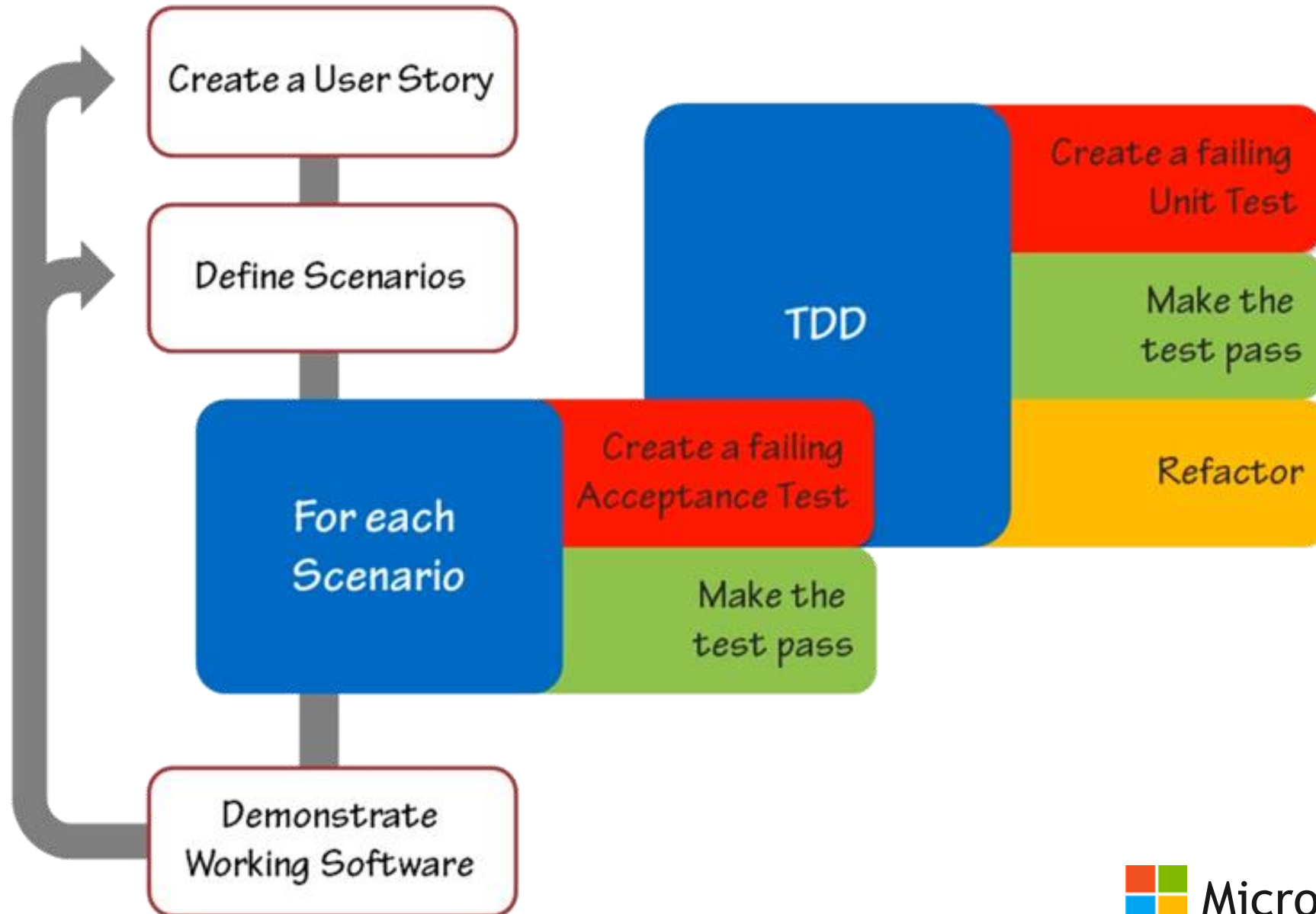
Shared team understanding of what is being built

Helps document what the system should do

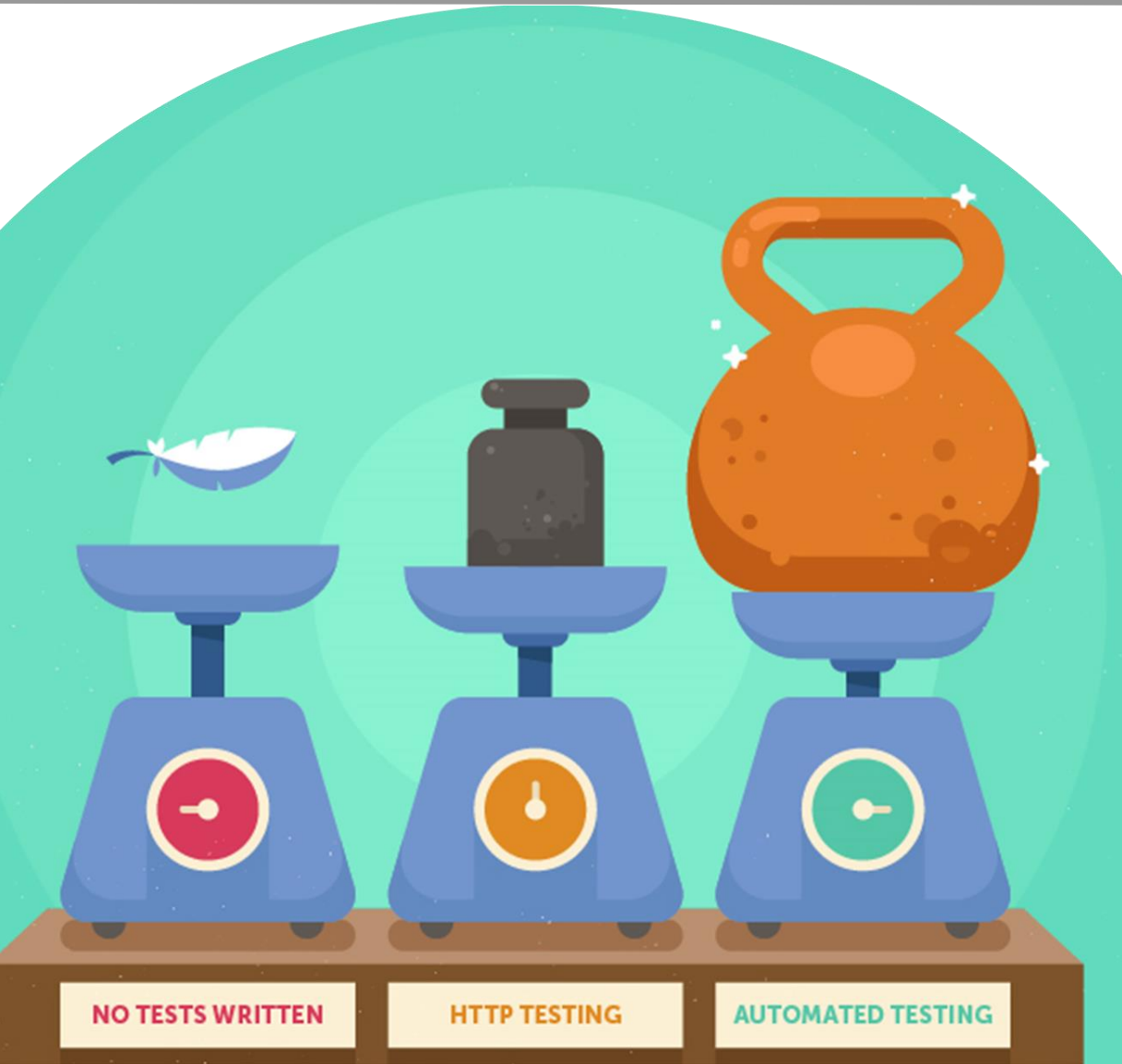
Helps define what "done" means



# BDD – From Ideas to working software



# Cucumber /SpecFlow - It is ok to not be Behaviour/Test Driven



Business Facing and user centric

Pass/fail Regression Automated Tests

Helps document the system's expectations

Much Better than no tests

Much Better than subcutaneous tests

# The Need for Manual Testing



There is no replacement  
for manual testing  
performed by humans.



Automated testing always  
going to be a subset what  
can be tested and cannot  
validate navigation,  
presentation, aesthetics or  
find hidden bugs.



However, collating and  
reporting on manual  
testing efforts is  
traditionally a challenging  
and tedious job.







hello world




ALL WORK IMAGES VIDEOS MAPS NEWS SHOPPING

3,010,000 Results Date Language Region

 Microsoft

 **Hello world**  
Distribution list  
✉ Helloworld2990@service.microso...

Members (3)

-  Alana Robinson  
SENIOR DESIGN MANAGER
-  Divya Srinivasan  
UX DESIGNER
-  Kirti Ahlawat  
DESIGNER 2

Show profile >

See all results from Microsoft >

## "Hello, World!" program - Wikipedia

[https://en.wikipedia.org/wiki/"Hello,\\_World!"\\_program](https://en.wikipedia.org/wiki/)

< Overview **Time to hello world** History Variations See also >

"Time to hello world" (TTHW) is the time it takes to author a "Hello, World!" program in a given programming language. This is one measure of a programming language's ease-of-use; since the program is meant as an introduction for people unfamiliar with the language, a more complex "Hello, World!" program may indicate that the programming language is less approachable. The concept has been extended beyond programming languages to APIs, as a measure of how simple it is for a new deve...

Wikipedia · Text under CC-BY-SA license

# Web UI Automation Testing

## "Hello, World!" program

Computer Program



A "HELLO, WORLD!" program generally is a computer program that outputs or displays the message "Hello, World!". Such a program is very simple in most programming languages, and is often used to illustrate the basic syntax of a programming lang...

Wikipedia

Data from: Wikipedia  
Wikipedia text under CC-BY-SA license

Suggest an edit

See results for

HelloWorld Inc (Company)

HelloWorld is a digital marketing solutions company.

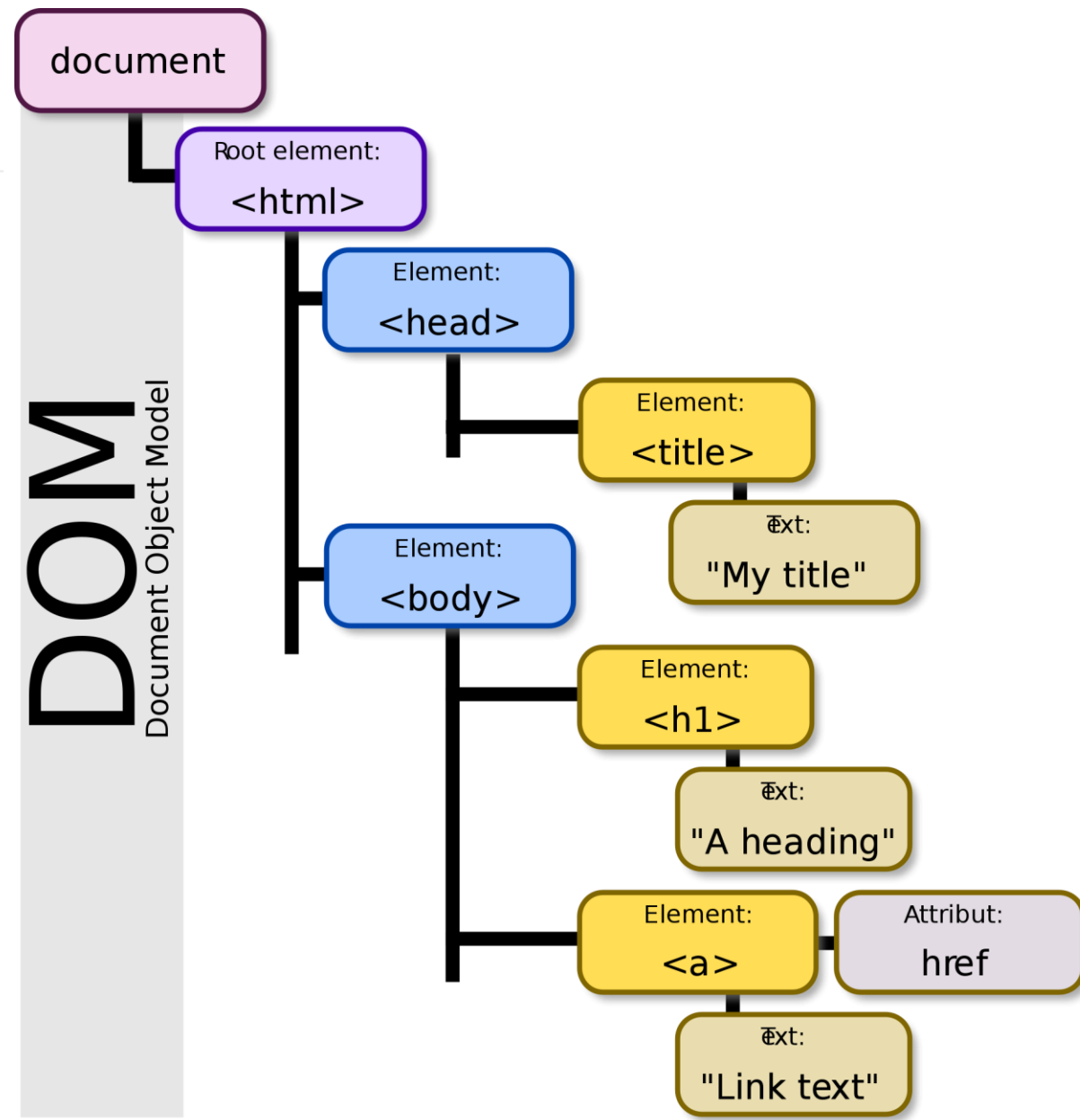
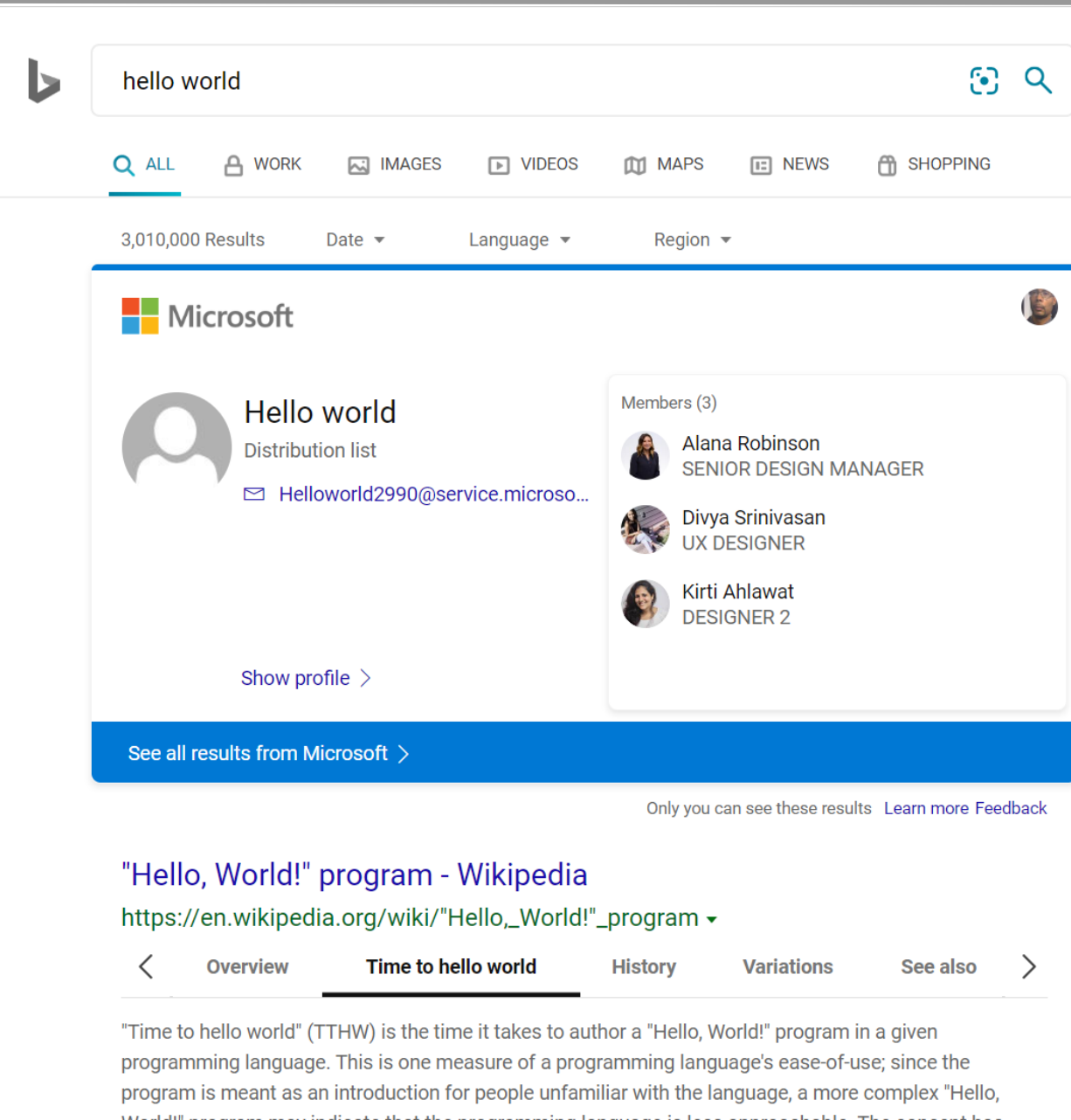
Hello World (2019 Film)

Hello World is a 2019 Japanese animated sci-fi comedy drama film directed by Tomohiko Itō and produced by...



Microsoft Services

# Anatomy of a Web Page



# Accessing Web Page elements

```
<html>
  <head>
    <title>This is a Document!</title>
    <link rel="stylesheet" href="main.css">
  </head>
  <body>
    <h1>This is a header!</h1>
    <p id="excitingText">
      This is a paragraph! <em>Excitement</em>
    </p>
    <p>
      This is also a paragraph, but it's not
      nearly as exciting as the last one.
    </p>
  </body>
</html>
```

```
#excitingText {
  color:red
}
```



```
document.getElementById("excitingText");
```



```
//*[@id="excitingText"]
```



```
#excitingText
```

```
public void IsRecurring()  
{  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
    Assert.IsTrue(customer.IsRecurringSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargedResultToFailOnceShouldStillBeCurrent()  
{  
    //Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    customer.RecordChargedResult(false);  
  
    // Assert  
    Assert.IsTrue(customer.HasCurrentSubscription);  
}
```

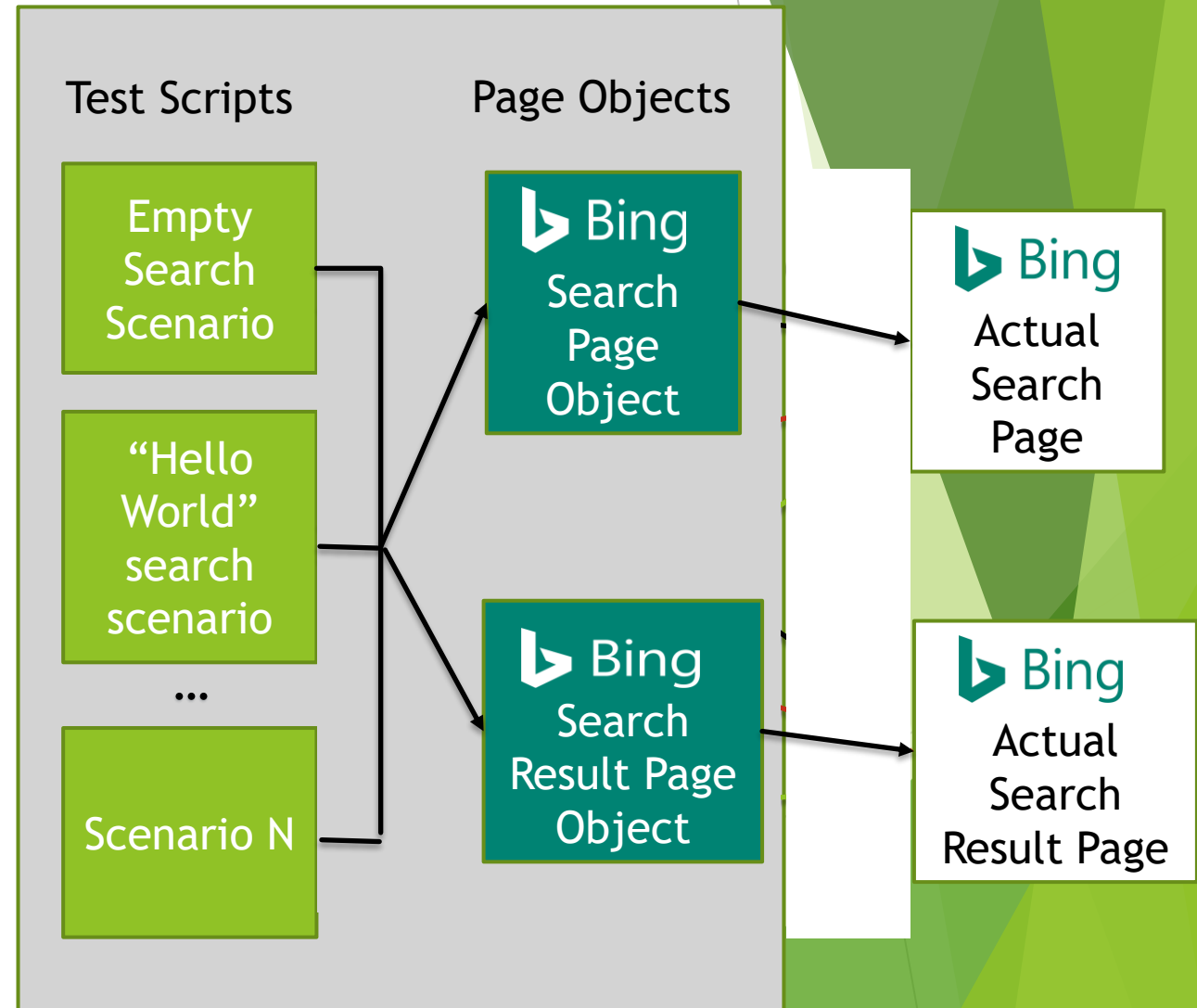
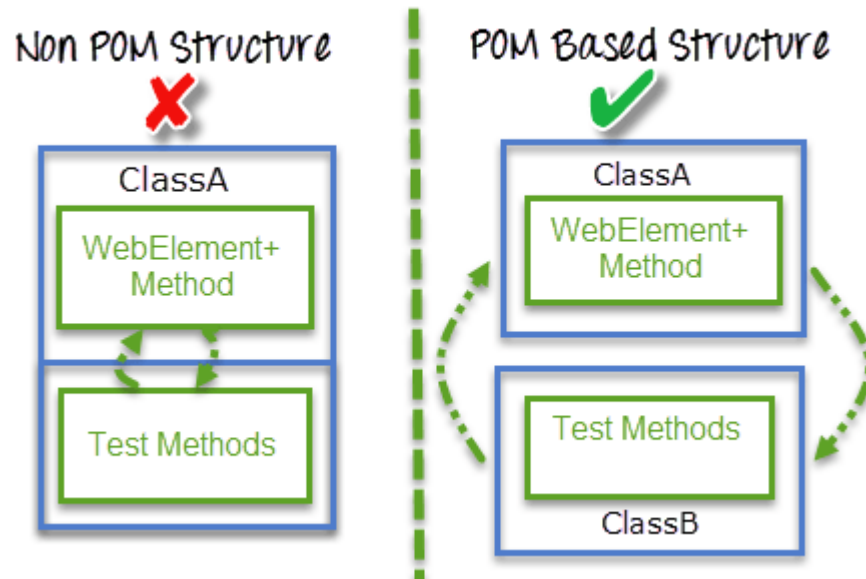
```
[Test]  
public void SubscriptionRecordChargeResultToFailMaximumTimesIsNoLongerCurrent()  
{  
    // Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    for (var i = 0; i < Customer.MAX_PAYMENT_FAILURES; i++)  
        customer.RecordChargedResult(false);  
  
    //Assert  
    Assert.IsFalse(customer.HasCurrentSubscription);  
}
```

# Exercise 2

## Finding elements with the Web Browser

# Page Object Model

- Design Pattern to create Object Repository for Web UI Elements.
- Abstract and encapsulates the responsibility into dedicated Page class to access or perform actions on its counterpart
- Page's actions either stay on the same or navigate to another page
- Makes code more readable, maintainable and reusable





```
public void IsRecurring()  
{  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
    Assert.IsTrue(customer.IsRecurringSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargedResultToFailOnceShouldStillBeCurrent()  
{  
    //Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    customer.RecordChargedResult(false);  
  
    // Assert  
    Assert.IsTrue(customer.HasCurrentSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargeResultToFailMaxTimesShouldNoLongerBeCurrent()  
{  
    // Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    for (var i = 0; i < Customer.MAX_PAYMENT_FAILURES; i++)  
        customer.RecordChargedResult(false);  
  
    //Assert  
    Assert.IsFalse(customer.HasCurrentSubscription);  
}
```

# Exercise 3

## Writing UI Automated with elenium and Pages Objects

tion 'FunctionalTests' (1 proje

FunctionalTests

Properties

References

Assertions

Configuration

Features

AddingToPlanesToCart.fe

AddingToPlanesToCart

MultipleBrowserTesting.fe

Models

Pages

Steps

Transformation

App.config

DummyUnitTest.cs

packages.config

Solution...

Team Exp...

Notificati...

Feature: Adding Planes To Cart

As a scale model enthusiast

I want to add paper or model planes to my cart

So that I can place an order

@Failing

Scenario: Adding the model "Fourth Coffee Flyer" airplane to the cart

Given I have selected the "Model Airplanes"

And I have selected the 1st plane

When I add the plane

Then I should be redirected to the cart page's url starting with "<http://tailspintoys.com/newwebsite>"

And there should be 1 plane(s) in the cart

#Next step fails because only the first title gets picked

And the page title should be "Fourth Coffee Flyer"

And the 1st plane's description in the cart should be "Fourth Coffee Flyer"

@Passing

Scenario: Adding a third paper "Wingtip Toys Stunt Plane" airplane to the cart

Given the cart has the following planes

Type	Description
Model Airplanes	Trey Research Rocket
Model Airplanes	Northwind Trader

And I have selected the "Paper Airplanes"

And I have selected the 2nd plane

When I add the plane


Then I should be redirected to the cart page's url starting with "<http://tailspintoys.com/newwebsite>"

And there should be 3 plane(s) in the cart

And the 3rd plane's description in the cart should be "Wingtip Toys Stunt Plane"


And the 3rd plane's quantity in the cart should be 1

And the 3rd plane's total price in the cart should be \$50.00

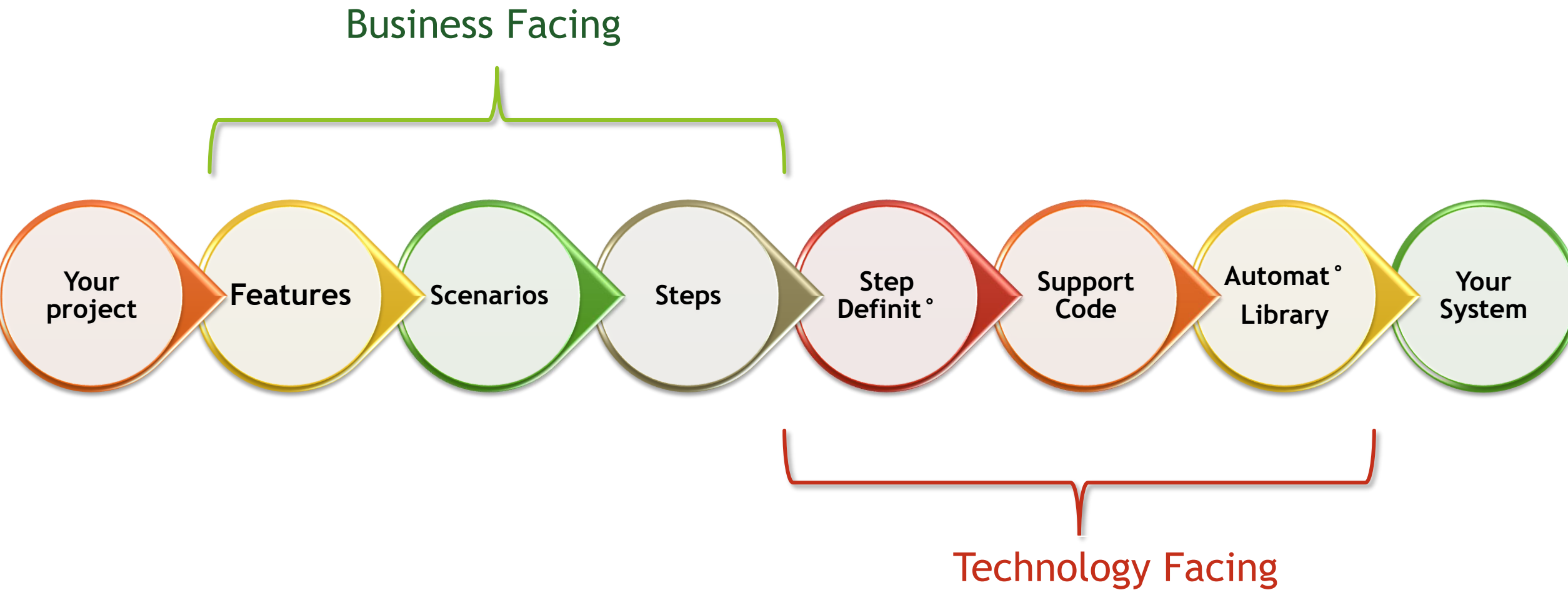


# SpecFlow

# Fundamentals

 Microsoft Services

# SpecFlow Testing Stack



# [Binding]



## Step Definition Attributes

- Can annotate single methods with multiple step definitions
- Can parameterise methods with regular expression
- Supports different styles, parameter matching & table matching



## Hooks

- perform additional automation logic before or after test run, feature, scenario, scenario block and step



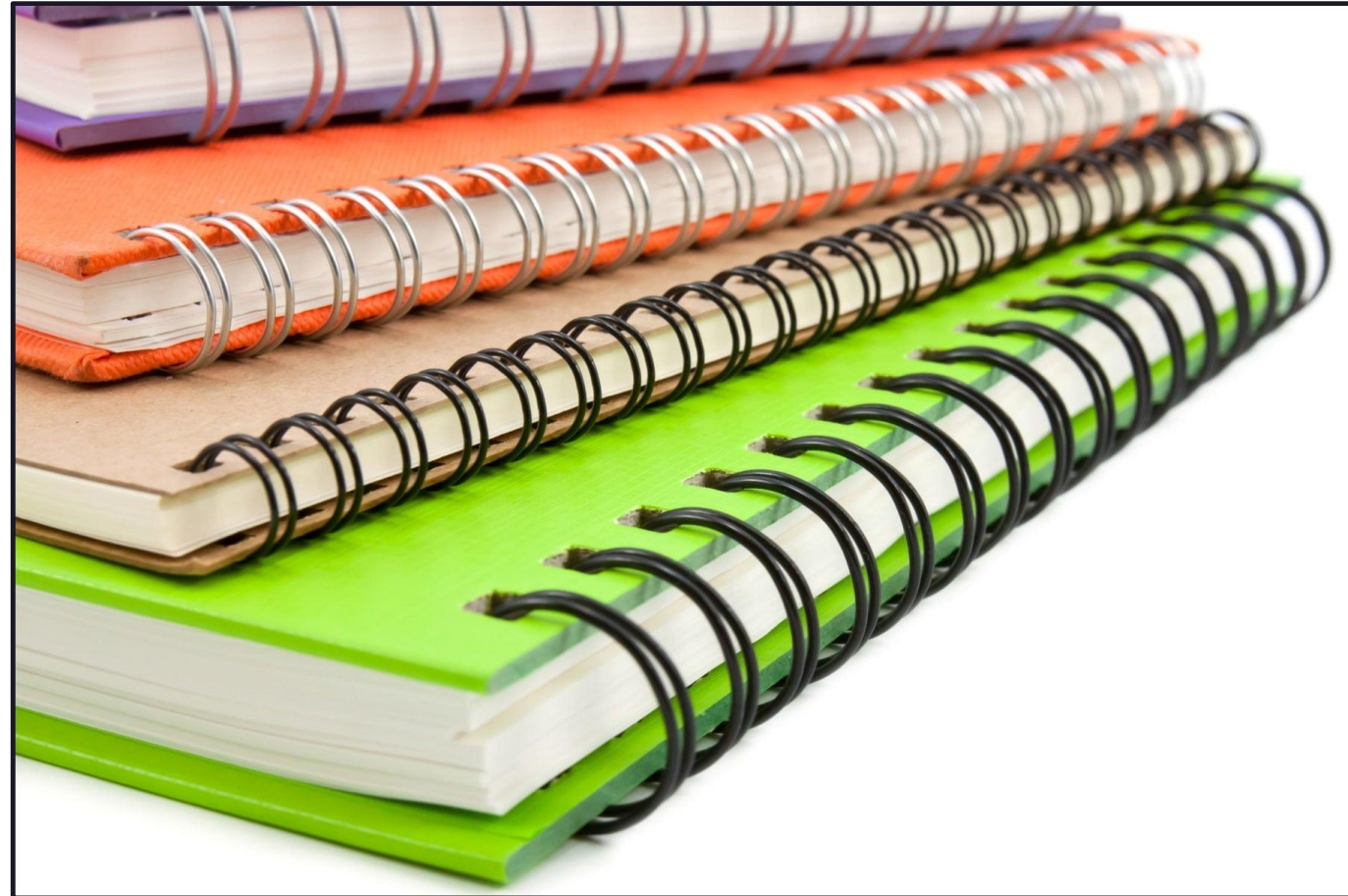
## Step Argument transformations

- Apply custom argument transformation in step definition methods from sting to a specified .NET type



## Scoped bindings

- Restricts Step definition and Hooks scope at the feature, scenario titles or tags
- Allows to define different automation logic



```
[BeforeScenario("web")]  
0 references | 0 changes | 0 authors, 0 changes  
public static void BeforeWebScenario()  
{  
    if (ScenarioContext.Current.ScenarioInfo.Tags.Contains("automated"))  
        StartSelenium();  
}
```



# Sharing Data Between Bindings



## Field Instance

- Share data between different steps of the same scenario class



## Context Injection

- Life span is limited to a scenario execution.
- Injected IDisposable objects are disposed after scenario execution.



## Feature Context

- FeatureContext.Current stores key/values persisted for the duration of the feature
- FeatureContext.FeatureInfo provides more information on the feature (title, tags, etc.)



## Scenario Context

- ScenarioContext.Pending prevents following steps from being executed
- ScenarioContext.TestError holds the last exception/error occurred
- ScenarioContext can also be injected







# Specflow



## Anti-patterns

# SpecFlow Anti-Patterns



## Beginner's mistakes

- Lots of user interface details
- Describing actions using personal pronoun
- No clear separation between Given/When/Then
- Multiple When



## No living documentation

- Doesn't describe an example of business rule or scenario
- Too many details or hard to tell what is being tested
- Not using narrative section of a Feature



## Bad collaboration

- Too high level
- Devs or testers writing scenario without talking to business (or vice-versa)
- Writing feature file after code is written



```
public void IsRecurring()  
{  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
    Assert.IsTrue(customer.IsRecurringSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargedResultToFailOnceShouldStillBeCurrent()  
{  
    //Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    customer.RecordChargedResult(false);  
  
    // Assert  
    Assert.IsTrue(customer.HasCurrentSubscription);  
}
```

```
[Test]  
public void SubscriptionRecordChargeResultToFailMaximumTimesIsNoLongerCurrent()  
{  
    // Arrange  
    var customer = new Customer { HasSubscription = true, SubscriptionType = SubscriptionType.Monthly };  
  
    // Act  
    for (var i = 0; i < Customer.MAX_PAYMENT_FAILURES; i++)  
        customer.RecordChargedResult(false);  
  
    //Assert  
    Assert.IsFalse(customer.HasCurrentSubscription);  
}
```

# Hands on Lab

## SpecFlow Bing Search Scenarios



Q&A

You have

Questions

We have

Answers



Microsoft Services

# Thank You

ευχαριστώ    Salamat Po    متشكراً    شكراً    Grazie  
благодаря    ありがとうございます    Kiitos    Teşekkürler    谢谢  
ໂພນດຸນດຣັບ    Obrigado    شكریه    Terima Kasih    Dziękuję  
Hvala    Köszönöm    Tak    Dank u wel    дякую    Tack  
Mulțumesc    спасибо    Danke    Cám ơn    Gracias  
多謝晒    Ďakujem    תודה    நன்றி    Děkuji    감사합니다