Specflow

* BDD uses Cucumber, Cucumber is a Ruby based tool.
* Specflow works like Cucumber for a .net project
* It is an open source tool.

BDD (userstories)

* User stories are not whole requirements they are instructions & desires to start a discussion.
* Given When Then is a BDD scenario, it is an illustration for requirements as they can be used to automate a test & is the documentation of behaviour and executable specifications
* Misunderstandings can come from user stories as we don’t know what we don’t know ( the unknown unknown)
* For example, is we see a user story referring to a new user. This new user can be someone who just registered, someone who hasn’t bought something. Someone who is registered but never had their session cached etc. We need clear details of how new this user is and everything In between. We need more information so there are no ambiguities
* Given When then are illustrating requirements.
* The examples work together to back up the user story.
* Examples can begin discussions to discover hidden requirements or rules

What is BDD

* Behaviour Driven Development
* Specification by examples
* Acceptance driven test development ADTD
* All concepts above are the same
* BDD is a process to help connect the different process bubbles
* Agile testing book touches on BDD to explain further
* Agile testing moves bug hunting to bug prevention
* Main route cause of bugs is from the requirements (56%)
* If we can spot holes in requirements before anything has been developed we will reduce bug frequency.
* We need to better understand requirements
* We need to better validate their correctness

Approach

* Discovery – Process of coming up with good examples ( taken from user stories)
* Formulation – Taking examples and adapting it to Given When Then
* Automation – Automating the testing using the scenarios
* Supplementary testing saves us time during regression
* Acceptance criteria – Rule
* First break down the rules
* Then create examples from these rules
* We are looking for checkable rules
* The goal is to collect as many examples until the team has a shared understanding of what the story is for.
* For each example we can derive more tests and exploratory tests around the example
* Example mapping is best done using the 3 amigos or some derivative there of so there’s someone from every role and we can work together with the PO, dev and tester.
* Example mapping is not a must have but a good way to collect examples and break down the user stories

Example Mapping

* Get the Given When Then
* Collate the rules – Verify the rules are correct (ask PO)
* Come up with examples derived from the rules
* Make the tests the examples we have collated
* Each scenario should be interesting and there should be a reason it exists
* Gherkin is the syntax
* Names come from Cucumber
* Problem V solution domain, the business works on the problem and the devs work on the solution
* GIVEN – this is the set up (what state do we need to be in to start testing)
* WHEN – This is the action that drives the result
* THEN – This is the result (the expected result) this is the decider on if the test has passed or not (Assertion)

Day 2 Notes using Specflow to automate

* The solutions folder within Specflow is for reference at a later date to see how and why we done what we done.
* Default login = login with default credentials
* Run time = test time
* The features on the right hand side of specflow in the solution explorer ( right hand side of visual) is the same layout as would be in Cucumber.
* Test Name is scenario name without spaces
* Specflow takes scenarios and makes them into tests
* Specflow CANNOT run the test. It can convert the scenarios to a testable test that you can run with a test runner ( resharper)
* WE have to tell Specflow which test runner to use for exa,ple Nunit – and we configure this is the App config file
* In the App.config file we can configure which runner we want to user (In the example of training we were using MStest)
* Specflow cant read English simply i.e.
  + “When I check the home page”
  + The above doesn’t mean anything to Specflow so we right click on the Given statement and we select “Generate step definitions”
  + The ‘Generate step defintiion’ shows you an example of what we can do.
* We need to name the class
* Sometimes we don’t need the Given example in the scenario
* Make step definitions reusable
* We need to call a parameter e.g. ( sting P0) – something understandable
* Don’t leave a ‘Pending’ statement – fix it with code ( right click and generate step definition )

Testing different layers of a site

C# notes

* A regular expression = (.\*)
  + A dot matches the character a star after the dot means we are saying ( any text in any length)
* Regular expressions can be very complex – but we want to keep it simple and understandable
* An @ sign excludes it from the double quote.
* You put a string in double quotes i.e
  + Var s1 =”Hello \t \rn “
* A double quote ends the string
  + @”C:temp”
  + You would use @ to get rid of the double quotes