Managing BDD

Test Case Management for BDD Automation

Agenda

- Brief Gherkin Walkthrough
- Technical Challenges
- Adopted Process and Workflow
- Gherkin Builder
- Implementation

Gherkin

What is Gherkin

- It is a Business Readable, Domain Specific Language that lets you describe software's behaviour without detailing how that behaviour is implemented
- Gherkin serves two purposes: documentation and automated tests
  - A bonus feature: notifies you of unimplemented test steps, with suggestions
- Gherkin's grammar is defined in the Treetop grammar that is part of the Cucumber codebase

Gherkin Syntax

- Gherkin is a line-oriented language that uses indentation to define structure
- Line endings terminate statements (eg, steps)
- Either spaces or tabs may be used for indentation
- Most lines start with a keyword
- Comment lines are allowed anywhere in the file
  - They begin with zero or more spaces, followed by a hash sign (#) and some amount of text
- A parser divides the input into features, scenarios and steps
  - When you run the feature the trailing portion (after the keyword) of each step is matched to a code block called Step Definitions

Gherkin Composition

- A ***Feature*** is a set of functionality
  - A single ***Feature*** is has its own file (ending in .feature)
  - Features are typically composed of multiple ***Scenarios***
- A ***Scenario*** is a block of statements that describe some desired behavior
  - ***Scenarios*** specify *What* and should avoid answering the question *How*
  - A ***Scenario Outline*** is a block of statements (***Scenario***) that gets repeated over a set of data
    - An ***Example*** table specifies input parameter data to allow reuse of test steps
- ***Background Steps*** may run before each ***Scenario*** to reduce redundancy in the ***Scenarios*** that make up the ***Feature***

Gherkin Composition

- A ***Scenario*** (test) consists of three parts:
- Given
  - The preconditions of the system under test
  - The setup of the systems state if you want
- When
  - The actual change of the system
  - Transforming it from the initial state to the final state
- Then
  - The expected final state of the system
  - The verification that the state change was the desired change

GWT Example

Given I have a new user who has completed basics

And I am logged in

When I access the dashboard

Then I validate lifestyle button unlocked

Given

- The purpose of givens is to put the system in a known state before the user (or external system) starts interacting with the system (in the *When* steps)
- Avoid talking about user interaction in givens
- If you were creating use cases, *Givens* would be your preconditions

Examples:

- Given I have a new registered user
- Given I am logged in

Bad Example:

- Given I login

When

- The purpose of *When* steps is to describe the key action the user performs
- ***Scenarios*** should limit the usage of *When*s to four or five steps
  - Look at testing and/or exercising one particular area of code to avoid cascading failures

Examples:

- When the state "California" is selected
- When I login
- When I click on the site pairing form
- When I fill out the overall health form to womens health

Then

- The purpose of *Then* steps is to observe outcomes
- The observations should be related to the business value/benefit in your feature description
- The observations should also be on some kind of output
    - Something that comes out of the system (report, user interface, message)
    - Not something that is deeply buried inside it (that has no business value)

Examples:

- Then I see the email error of "Please enter a valid email address"
- Then I see the consent form
- Then I can re-run the refresher consent form

Gherkin Example

```
@Feature_AC-21362 @dashboard @pmi @subscriber
Feature: Dashboard
  As a user
  I want to have access to the dashboard
  So that I can see and fill out forms

  @AC-21422 @unlock-forms
  Scenario: Lifestyle and Overall Forms section appearance changes in Dashboard upon completing Basics
    Given I have a new user who has completed basics
    When I login
    When I access the dashboard
    Then I validate lifestyle button unlocked
    Then I validate overall button unlocked

  @AC-21491 @complete-forms
  Scenario Outline: Completing a form shows as completed
    Given I have a new user who has completed "<form>"
    When I login
    When I access the dashboard
    Then the <form> form will show as completed

    Examples:
      |form|
      |basic|
      |overall|
      |lifestyle|
      |ehr|
```

Technical Challenges

Technical Challenges

- Distributed team with different skill sets all ‘want to’ write tests
- Test cases still require source control
- Need one source of truth
- Growing test step library can quickly become unwieldy
- No common supporting tooling

Process & Workflow

Storage

- All test cases stored in GIT
  - *Source of truth*
- Features are represented as Epics in JIRA
  - A unique tag exists for each feature representing JIRA epic key
- Scenarios are represented as Tests in JIRA
  - A unique tag exists for each scenario representing JIRA test key
- Every time a test case is executed with the JIRA flag, JIRA is updated
  - All test information is updated, including title, steps, and links
  - A test cycle is generated which records status of test execution

Problem

- Git is a technical tool
- How can we support non-technical people writing tests
- How can non-technical people easily edit tests

Automated Tests are code - and need to be treated as such

Gherkin Builder

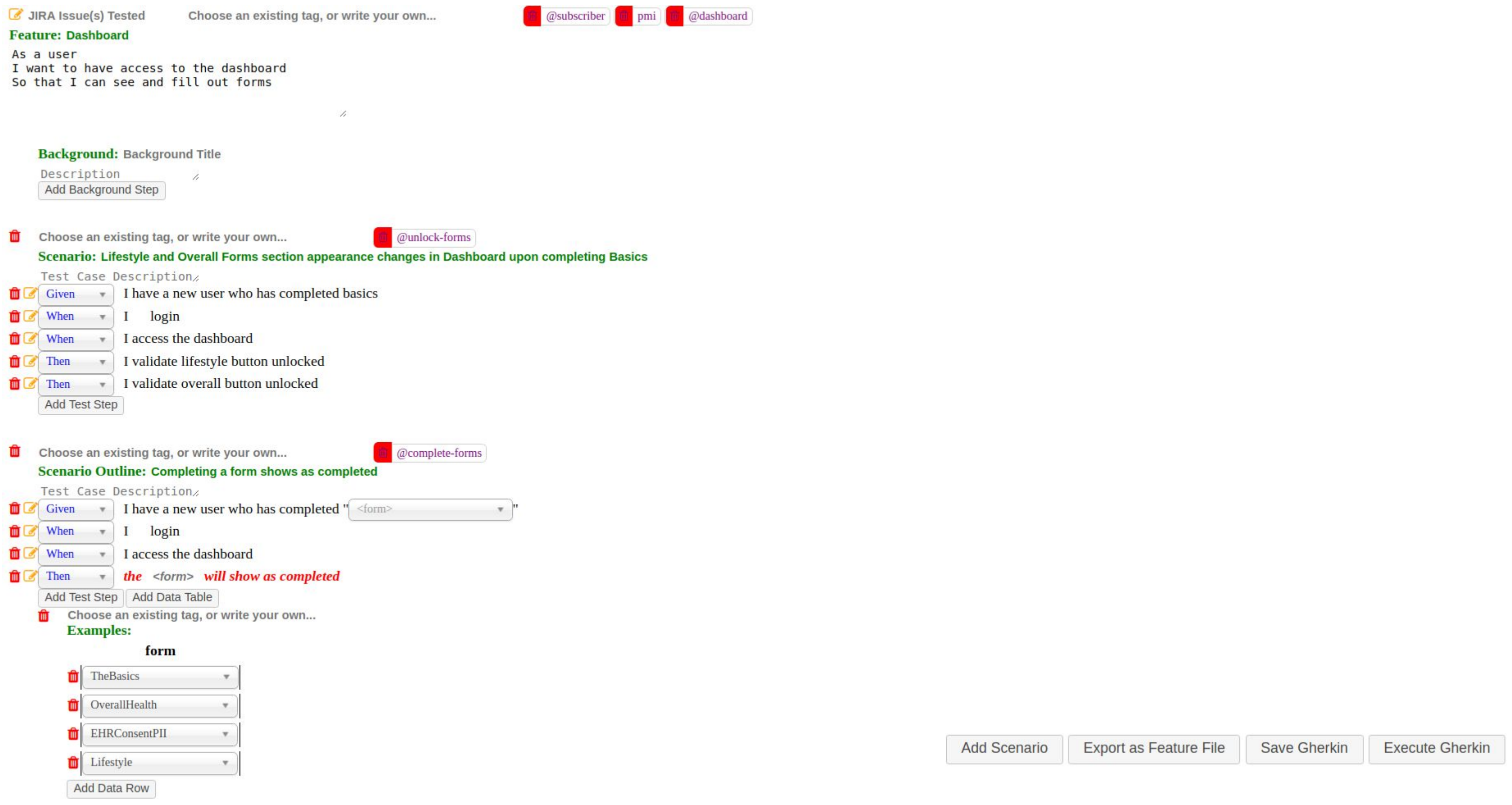
Gherkin Builder

- Gherkin test steps quickly grow
- The un-initiated often don’t know what steps exist
- Easy for similar/repetitive steps to appear in ‘code base’
- Few good non-technical tools for managing Gherkin steps

Gherkin Builder

- Provides simple structuring and auto-completion for writing Gherkin tests
- Suggests test steps based on already implemented test steps
  - Also suggests tags based on existing tagging
- Write a ***Scenario*** and easily turn it into a ***Scenario Outline***
- JIRA integration
  - Create a new Feature, or add tests to an existing Feature in JIRA
  - Create links between JIRA dev stories and tests
  - Exports tests directly into Zephyr
- External Tooling support
  - Provides ability to link execution or other capabilities to tools such as Jenkins

Demo



JIRA Issue(s) Tested   Choose an existing tag, or write your own...   @subscriber pmi @dashboard

Feature: Dashboard

```
As a user
I want to have access to the dashboard
So that I can see and fill out forms
```

Background: Background Title

Description

Add Background Step

Choose an existing tag, or write your own...   @unlock-forms

Scenario: Lifestyle and Overall Forms section appearance changes in Dashboard upon completing Basics

Test Case Description

Given I have a new user who has completed basics
When I login
When I access the dashboard
Then I validate lifestyle button unlocked
Then I validate overall button unlocked

Add Test Step

Choose an existing tag, or write your own...   @complete-forms

Scenario Outline: Completing a form shows as completed

Test Case Description

Given I have a new user who has completed "<form>"
When I login
When I access the dashboard
Then *the <form> will show as completed*

Add Test Step Add Data Table

Choose an existing tag, or write your own...

Examples:

form
TheBasics
OverallHealth
EHRConsentPII
Lifestyle

Add Data Row

Add Scenario Export as Feature File Save Gherkin Execute Gherkin

Implementation

Writing Tests

IDE

- Write the tests using autocomplete available in IDEs
- Create empty JIRA test and capture issue key
- Add issue key as tag to Gherkin test case
- Add JIRA links using @tests-XX-XXXX format
- Commit using typical git workflows

Gherkin Builder

- Write test case using hosted tool
- Add JIRA links and tags
- Use publish to JIRA option

Editing Tests

IDE

- Checkout latest code from git
- Make updates to test case
- Commit using typical git workflows

JIRA

- Use special JIRA field to edit test case
- Routed through Jenkins to Gherkin Builder
- Make updates to test case
- Use publish to JIRA option

Executing Tests

Locally

- Checkout latest code from git
- Execute code from command line or IDE
    - Remember to include JIRA flag for updates if desired

JIRA

- Use special JIRA field to edit test case
- Taken to custom Jenkins job for test execution
- Navigating back to JIRA shows test case execution

Implementation

https://github.com/Coveros/GherkinBuilder

- Gherkin Builder code base consists of two parts
  - Glue Code Parser
    - Maven project
    - Scans provided folder for regular expressions
    - Builds javascript file containing possible test steps to be consumed
    - Support for multiple input types
  - Web App
    - PHP Project
    - Front end builder, using jquery to build Feature files
    - APIs for interacting with JIRA APIs and ZAPI
- Nightly build executes maven project against latest test automation code
  - Pushes any js/php updates and new test steps JS file to gherkin builder server

Questions?