Behavioral-Driven Development

Table of Contents

1. Tests

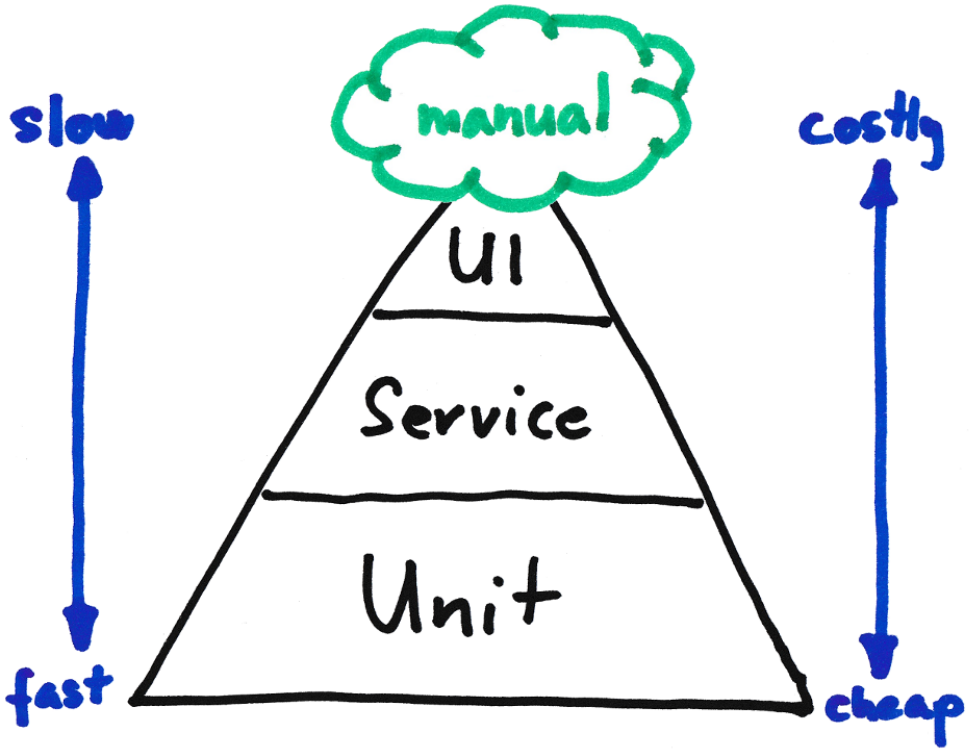


Figure 1. The testing pyramid (source here)

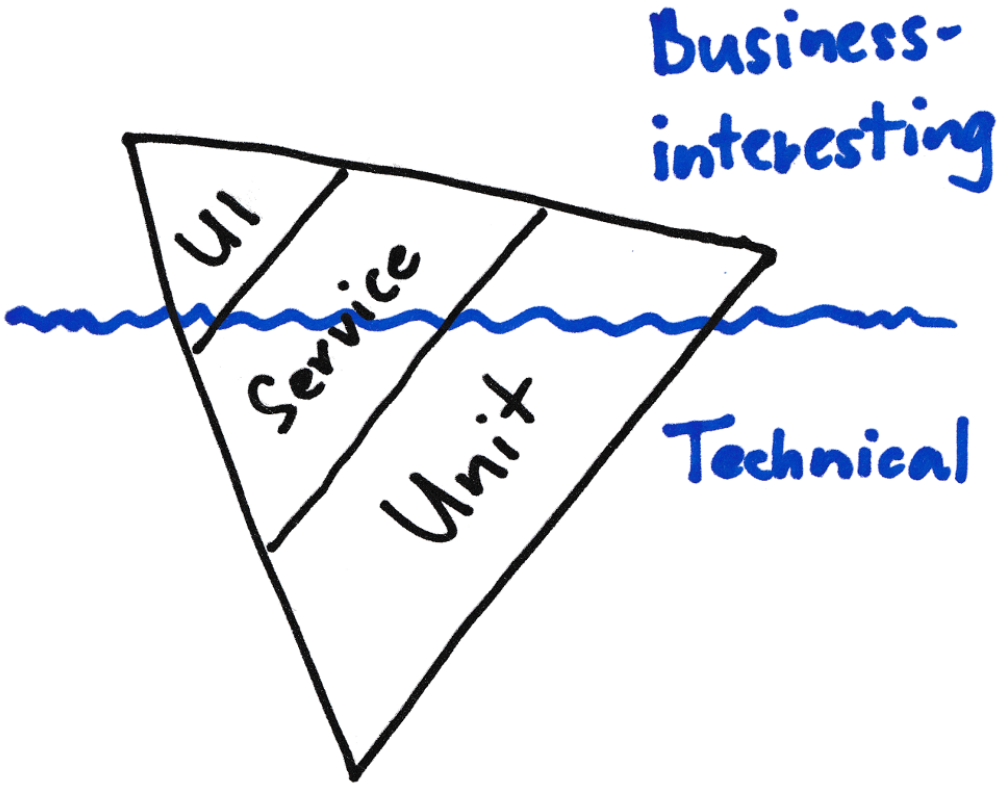


Figure 2. The testing pyramid at work (source here)

2. Features

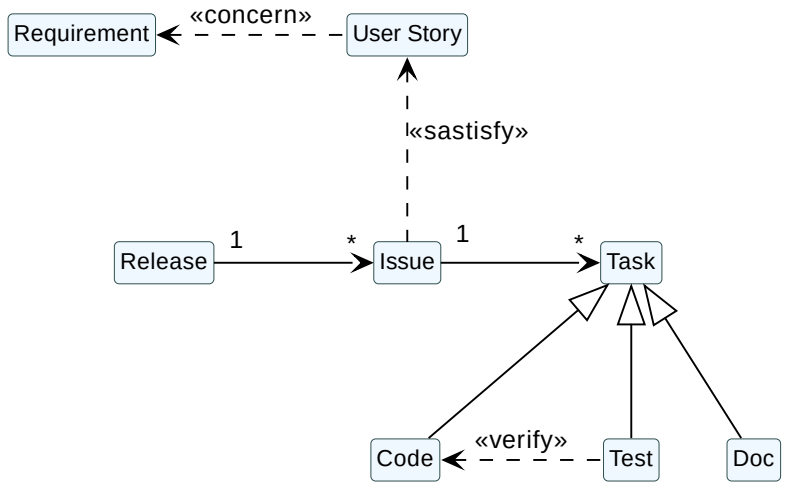


Figure 3. Artifacts traceability (Inspired from here)

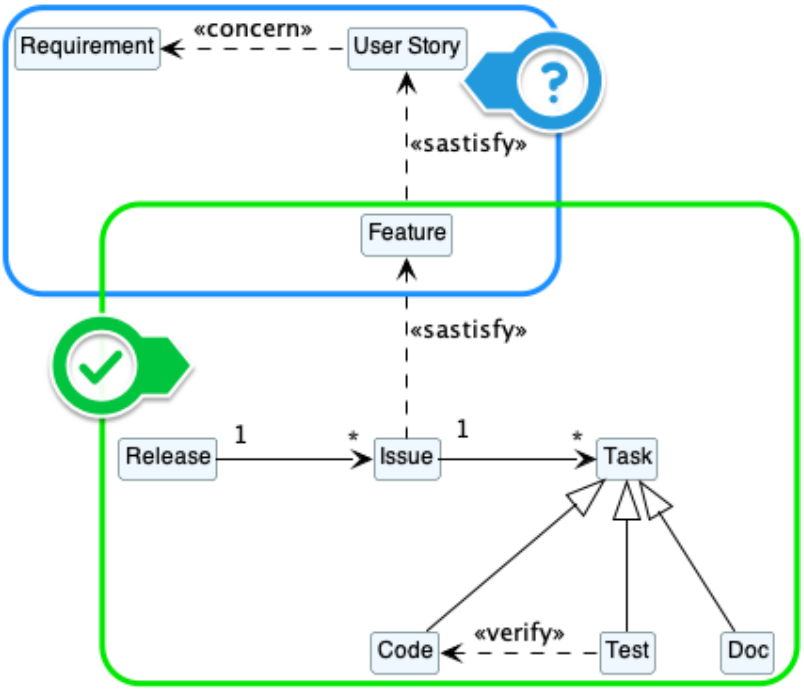


Figure 4. One more artifact

3. Principles

- Given
- When
- Then
- And

4. Examples

4.1. Some review activity (No code)

```
Feature: Reviewing a Ph.D. Thesis
    Every PhD thesis review has some recurrent steps

Scenario: A reviewer, being an expert on the field, should be cited somewhere
    Given A PhD thesis to review
    And a reviewer John Smith
    Then The thesis should cite John Smith's work

Scenario: A Ph.D. thesis should be an original work
    Given A PhD thesis to review
    And a reviewer John Smith
    Then The PhD.pdf should be checked agains plagiarism
```

Figure 5. Review PhD feature

4.2. This course materials! (No code)

```
#------------------------------
# Checking URLs
# JMB - 2020
#------------------------------
# language: en
Feature: Teaching Materials Quality Assemsment
    Every material should have correct links

Scenario: The URLs mentioned in an Asciidoc document should be verified (non 404)
    Given An Asciidoc file
    Then All the URLs should be active
```

Figure 6. Check URLs feature

4.3. Android example

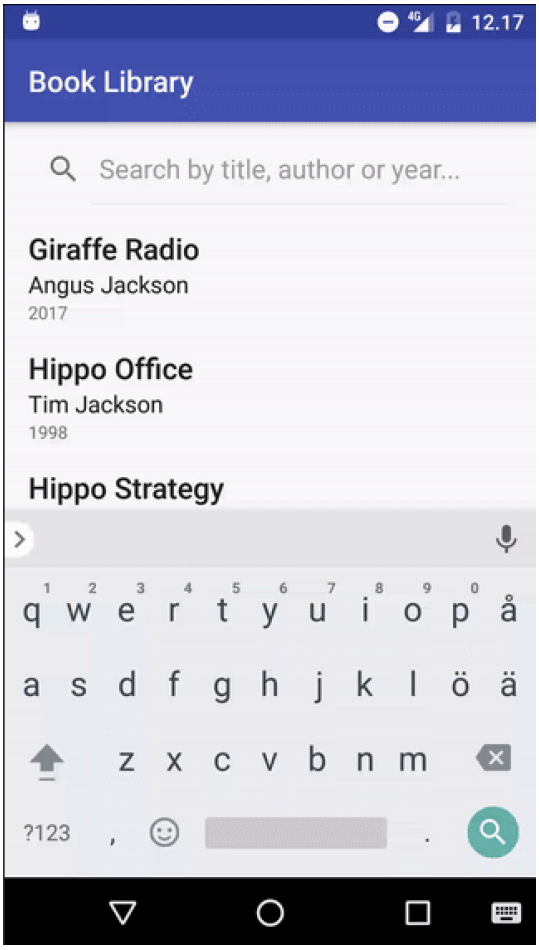


Figure 7. An Android app (source here)

A feature for this app (source here)

```
Feature: Book Search
    Scenario: Search books by author
      Given there's a book called "Tips for job interviews" written by "John Smith"
        And there's a book called "Bananas and their many colors" written by "James
Doe"
        And there's a book called "Mama look I'm a rock star" written by "John Smith"
      When an employee searches by author "John Smith"
      Then 2 books should be found
        And Book 1 has the title "Tips for job interviews"
        And Book 2 has the title "Mama look I'm a rock star"
```

4.4. Angular example

A "Getting Started" app

https://github.com/petermorlion/angular-getting-started

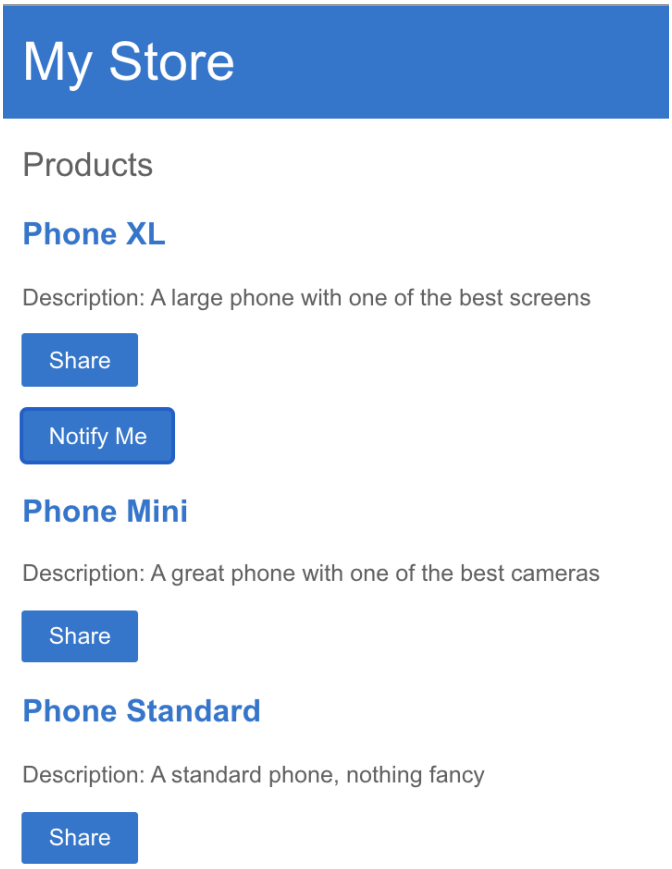


Figure 8. An Angular example app

Some feature description

```
Feature: Automatic discounts for premium customers
    Premium customers should automatically get a
    discount of 10% on purchases over $100.

    Scenario: Purchase over $100
        Given a premium customer
        And an order containing
            | item   | amount | price |
            | pencil | 100    | 1     |
            | paper  | 1      | 35    |
        When the customer checks out
        Then the total price should be 121.5
```

Figure 9. Some feature example

Running Cucumber (fail)

```
[→  /tmp/angular-getting-started git:(master) ✗ cucumber features/products-list.feature
Feature: Products List

  Scenario: Load the products list    # features/products-list.feature:3
    When we request the products list # features/products-list.feature:4
    Then we should receive            # features/products-list.feature:5
      | name           | description                                |
      | Phone XL       | A large phone with one of the best screens |
      | Phone Mini     | A great phone with one of the best cameras |
      | Phone Standard | A standard phone, nothing fancy            |

1 scenario (1 undefined)
2 steps (2 undefined)
0m0.051s

You can implement step definitions for undefined steps with these snippets:

When('we request the products list') do
  pending # Write code here that turns the phrase above into concrete actions
end

Then('we should receive') do |table|
  # table is a Cucumber::MultilineArgument::DataTable
  pending # Write code here that turns the phrase above into concrete actions
end


  Share your Cucumber Report with your team at https://reports.cucumber.io

  Command line option:    --publish
  Environment variable:   CUCUMBER_PUBLISH_ENABLED=true
  cucumber.yml:           default: --publish

  More information at https://reports.cucumber.io/docs/cucumber-ruby

  To disable this message, specify CUCUMBER_PUBLISH_QUIET=true or use the
  --publish-quiet option. You can also add this to your cucumber.yml:
  default: --publish-quiet
```

Figure 10. First run of the tests

Write Steps definitions

This is code linking the assertions with the running code.

Steps definition

```
import { When, Then } from 'cucumber';

When('we request the products list', function () {
    // Write code here that turns the phrase above into concrete actions
    return 'pending';
});

Then('we should receive', function (dataTable) {
    // Write code here that turns the phrase above into concrete actions
    return 'pending';
});
```

Running Cucumber (pass)

```
...

1 scenario (1 passed)
2 steps (2 passed)
0m05.270s
```

Figure 11. New run of the tests

5. Supported languages

Ruby (origin), Java, JavaScript,

- IntelliJ Cucumber for Java plugin
- Cucumber Eclipse plugin

6. Build

Make sure to integrate BDD in your build.

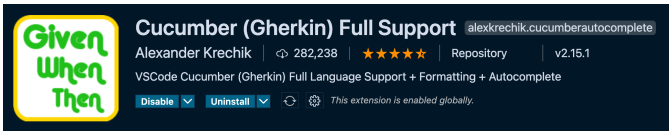
For maven/gradle, see https://cucumber.io/docs/tools/java/#build-tools

7. Find a plugin for your IDE

```
Feature: Products List

    Scenario: Load the products list
        When we request the products list
        Then we should receive
            | name           | description                                |
            | Phone XL       | A large phone with one of the best screens |
            | Phone Mini     | A great phone with one of the best cameras |
            | Phone Standard | A standard phone, nothing fancy            |
```

Figure 12. Cucumber plugin in use in VS Code

Example for VS Code: https://github.com/alexkrechik/VSCucumberAutoComplete



8. Gherkin tips

8.1. Avoid duplication

```
Scenario: Bad steps definition
    Given I go to the home page
    Given I check the about page of the website
    Given I get the contact details
```

```
Scenario: Good steps definition
    Given I go to the {} page
```

Sometimes you want to relax your language, to make it flow better.

```
Scenario: Alternative text
    Given I have {int} cucumber(s) in my belly/stomach
# Would match all these expressions:
#     I have 1 cucumber in my belly
#     I have 1 cucumber in my stomach
#     I have 42 cucumbers in my stomach
#     I have 42 cucumbers in my belly
```

Generic Step definition

```
@Given("I go to the {string} page")
public void i_want_to_open_page(String webpage) {
  webpageFactory.openPage(webpage);
}
```

8.2. Grouping step definitions

One file for each major domain object.

Example of step definition files

```
EmployeeStepDefinitions.java
EducationStepDefinitions.java
ExperienceStepDefinitions.java
AuthenticationStepDefinitions.java
```

8.3. Scenario outline (template)

```
Scenario Outline: eating
  Given there are <start> cucumbers
  When I eat <eat> cucumbers
  Then I should have <left> cucumbers

  Examples:
    | start | eat | left |
    |    12 |   5 |    7 |
    |    20 |   5 |   15 |
```

8.4. Language support

Over 70 languages!

Don't forget the `# language: fr`!

```
# language: fr
Fonctionnalité: Servir du café
  Afin de gagner de l'argent
  Les clients doivent être capables
  d'acheter du café à toutes heures

  Scénario: Acheter le dernier café
    Etant donné qu'il reste 1 café dans la machine
    Et que j'ai mis 1 dollar
    Quand j'appuie sur le bouton de la machine
    Alors je devrai recevoir un café
```

Figure 13. Example of feature in French (source)

9. How-to

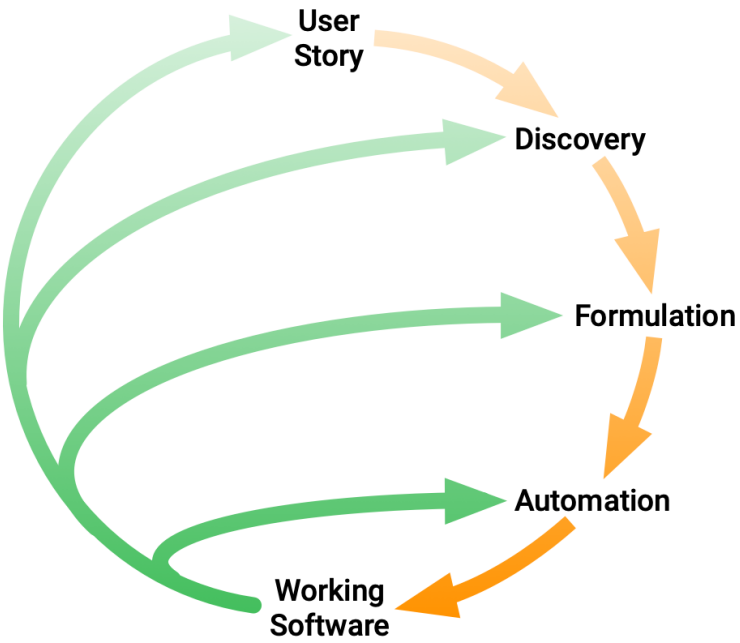


Figure 14. Discovery, Formulation and Automation (Source here)

10. Concepts

10.1. Feature

What follows is a basic description or name of the feature being tested or documented.

10.2. Rule

- represent a business rule that should be implemented
- used to group together several scenarios

New concept (since Gherkin v6)

10.3. Background

Steps that will be ran before every scenario in the feature.

10.4. Scenario

Name or basic description of a particular scenario testing the feature.

10.5. Scenario Outline

The scenario will run N times for every argument listed in examples explicitly passed by column name wrapped in angled brackets.

A Scenario Outline must contain one or more Examples

10.6. Examples

The list of static arguments that will be passed into a scenario outline.

10.7. The AAA pattern

```
// arrange
var component = ComponentFixture<UserUploadComponent>;
var user = new User(component);
// act
user.Save();
// assert
mock.Received.SomeMethod();
```

10.8. Given

A given step, or precondition that is assumed before continuing.

In the Arrange, Act, Assert paradigm, given represents "Arrange".

10.9. When

A when step, or the behavior that is to be asserted against.

In the Arrange, Act, Assert paradigm, given represents "Act".

10.10. Then

A then step, or in other words, the step in which a behavior's result is validated.

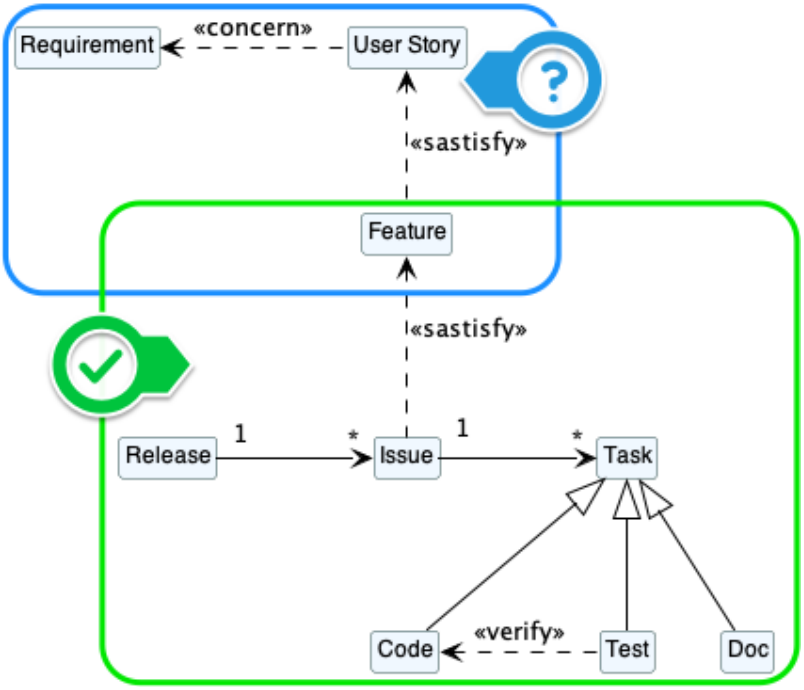
In the Arrange, Act, Assert paradigm, given represents "Assert".

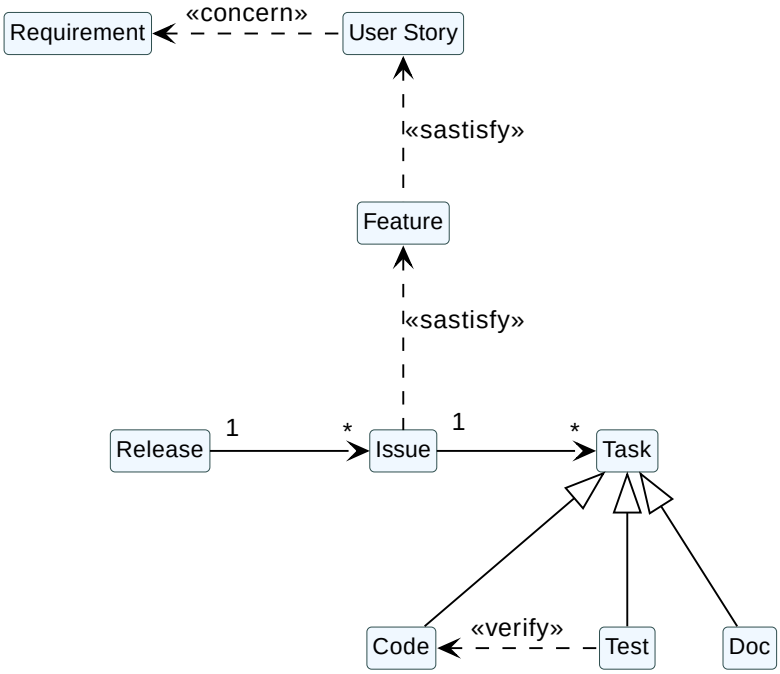
10.11. And

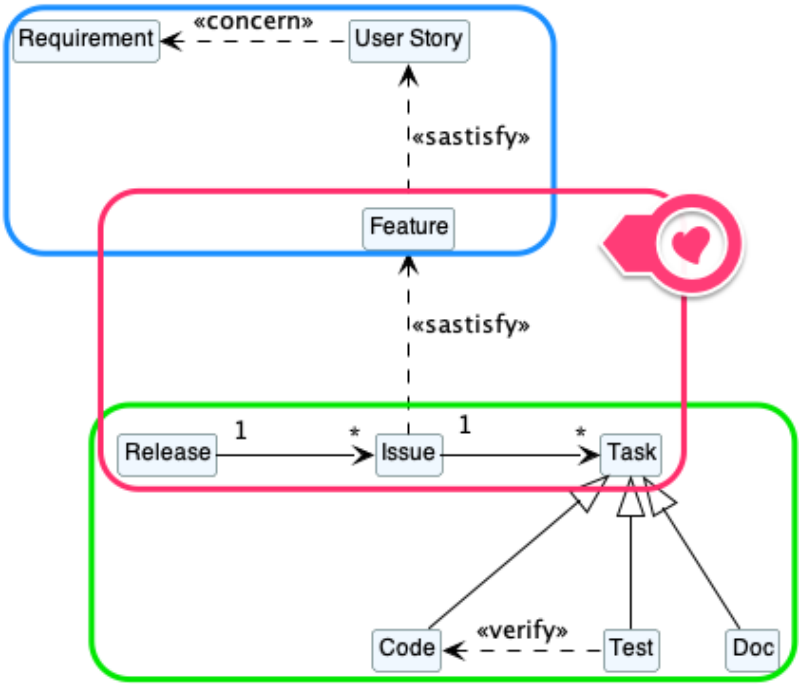
Used in conjunction with any of the keywords above.

If you have two given statements, instead of explicitly calling Given twice, you can say, " Given A And B".

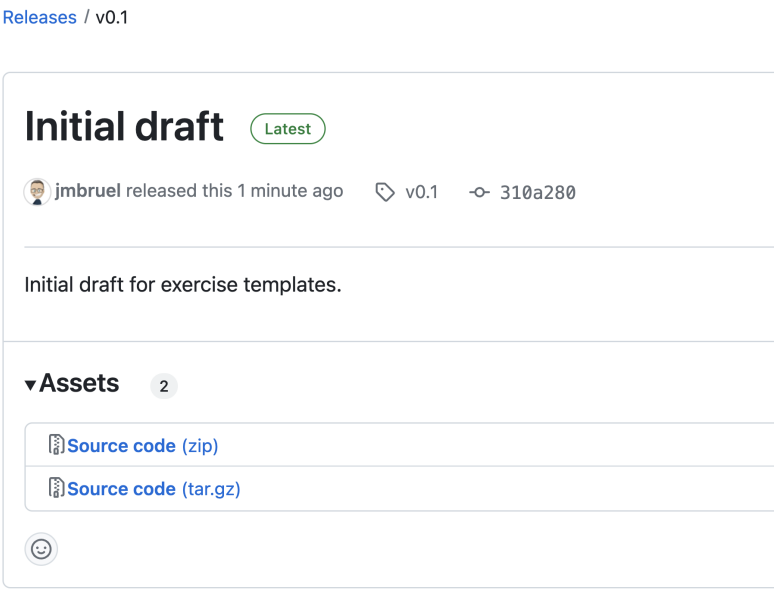
11. Links between US and feature



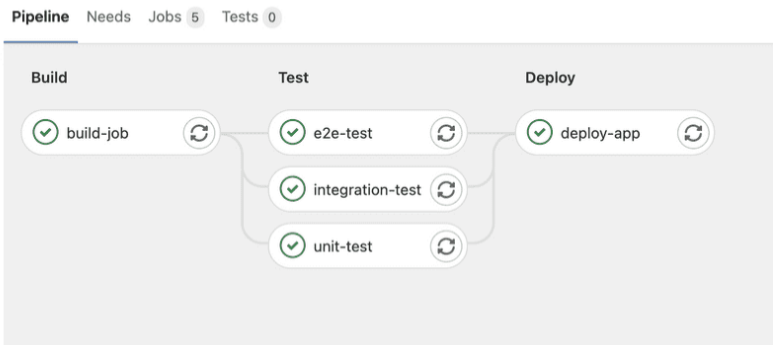




12. Links between Features and their implementation



13. CI/CD and BDD



14. BDD vs. TDD

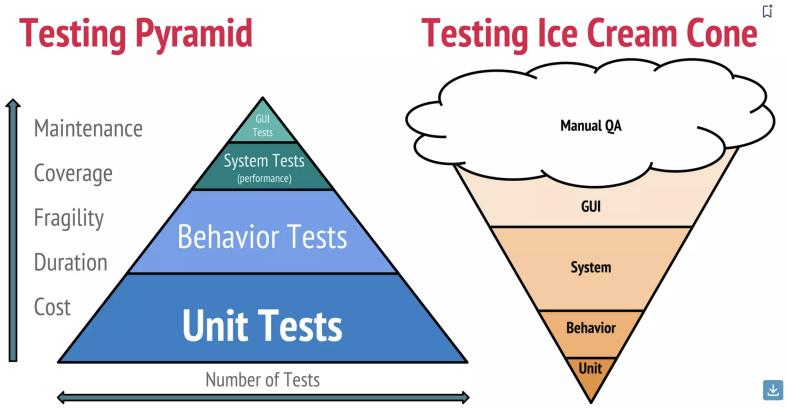


Figure 15. (source here)

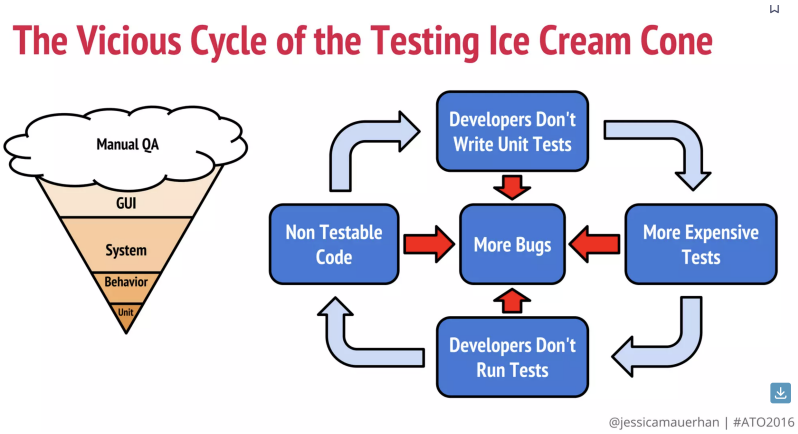


Figure 16. (source here)

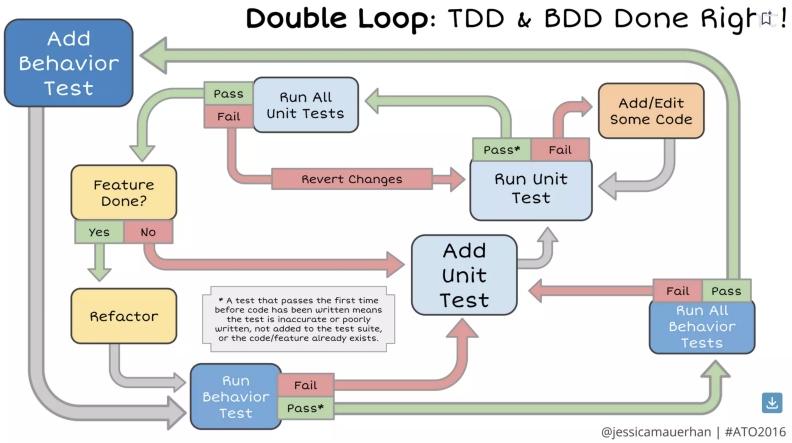


Figure 17. (source here)

15. Not just for code!

```
#-----------------------------
# Checking URLs
# JMB - 2020-2023
#-----------------------------
# language: en
Feature: Teaching Materials Quality Assemsment
    Every material should have correct links

Scenario: The URLs mentioned in an Asciidoc document should be verified (non 404)
    Given An Asciidoc file
    Then All the URLs should be active
```

Build tools

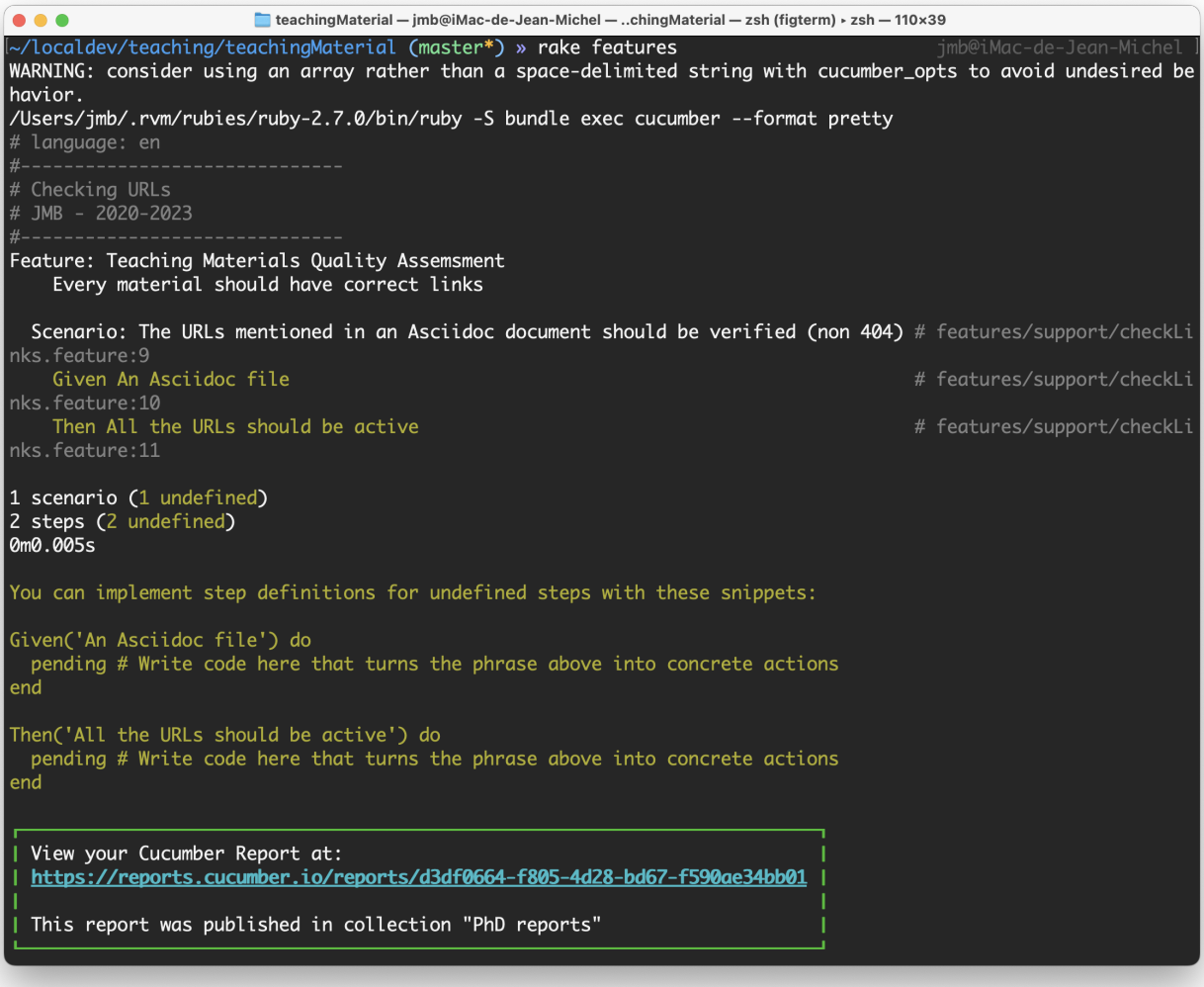
Cucumber can be run in several ways. Be aware that `rake cucumber`, `cucumber features`, and `autotest` with `E`

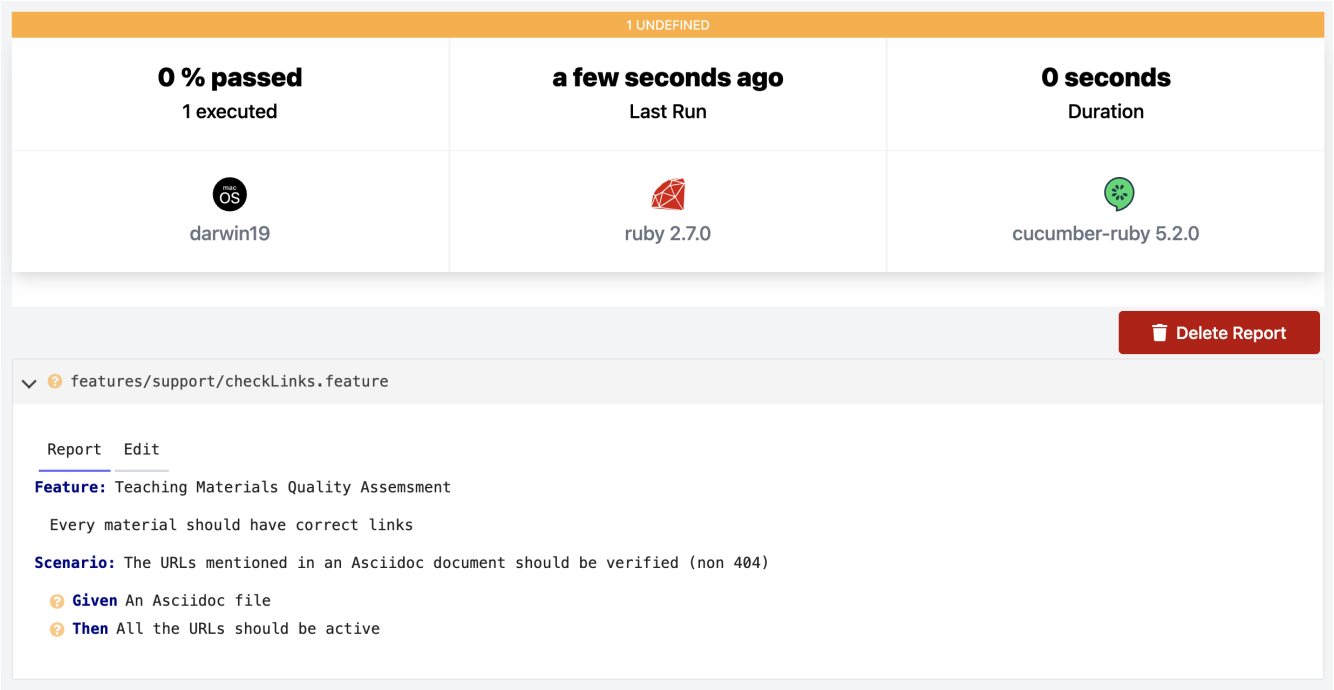
Rake

Running `rake cucumber` from the command line provides the simplest method to run Cucumber tests.

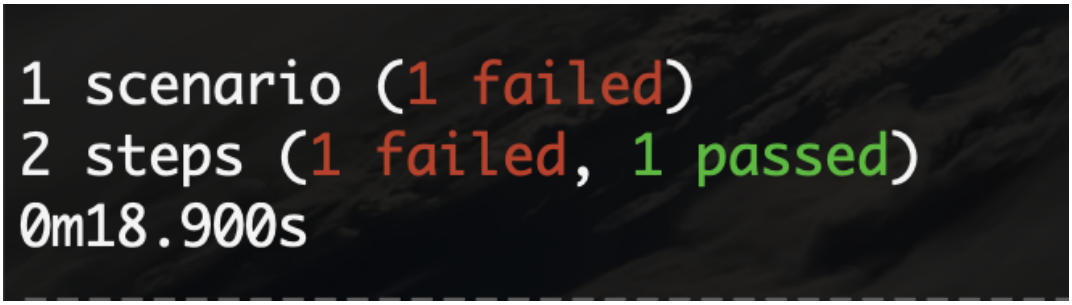
Using Rake requires a `Rakefile` with a `features` task definition. For example:

```
require 'rubygems'
require 'cucumber'
require 'cucumber/rake/task'

Cucumber::Rake::Task.new(:features) do |t|
  t.cucumber_opts = "--format pretty" # Any valid command line option can go here.
end
```


teachingMaterial — jmb@iMac-de-Jean-Michel — ..chingMaterial — zsh (figterm) ▸ zsh — 110×39
~/localdev/teaching/teachingMaterial (master*) » rake features
jmb@iMac-de-Jean-Michel
WARNING: consider using an array rather than a space-delimited string with cucumber_opts to avoid undesired behavior.
/Users/jmb/.rvm/rubies/ruby-2.7.0/bin/ruby -S bundle exec cucumber --format pretty
language: en
#-----------------------------
Checking URLs
JMB - 2020-2023
#-----------------------------
Feature: Teaching Materials Quality Assemsment
Every material should have correct links
Scenario: The URLs mentioned in an Asciidoc document should be verified (non 404) # features/support/checkLinks.feature:9
Given An Asciidoc file # features/support/checkLinks.feature:10
Then All the URLs should be active # features/support/checkLinks.feature:11
1 scenario (1 undefined)
2 steps (2 undefined)
0m0.005s
You can implement step definitions for undefined steps with these snippets:
Given('An Asciidoc file') do
pending # Write code here that turns the phrase above into concrete actions
end
Then('All the URLs should be active') do
pending # Write code here that turns the phrase above into concrete actions
end
View your Cucumber Report at:
https://reports.cucumber.io/reports/d3df0664-f805-4d28-bd67-f590ae34bb01
This report was published in collection "PhD reports"


1 UNDEFINED
0 % passed
1 executed
a few seconds ago
Last Run
0 seconds
Duration
darwin19
ruby 2.7.0
cucumber-ruby 5.2.0
Delete Report
features/support/checkLinks.feature
Report
Edit
Feature: Teaching Materials Quality Assemsment
Every material should have correct links
Scenario: The URLs mentioned in an Asciidoc document should be verified (non 404)
Given An Asciidoc file
Then All the URLs should be active

```ruby
Given('An Asciidoc file') do
    $source = "README.adoc"
  end

Then('All the URLs should be active') do
  system("asciidoc-link-check #{$source} -c config.json")
end
```




```
teachingMaterial — noglob rake features — features — zsh (figterm) ▸ node — 127×55
~/localdev/teaching/teachingMaterial (master*) » rake features                    jmb@iMac-de-Jean-Michel
WARNING: consider using an array rather than a space-delimited string with cucumber_opts to avoid undesired behavior.
/Users/jmb/.rvm/rubies/ruby-2.7.0/bin/ruby -S bundle exec cucumber --format pretty
# language: en
#-----------------------------
# Checking URLs
# JMB - 2020-2023
#-----------------------------
Feature: Teaching Materials Quality Assemsment
    Every material should have correct links

  Scenario: The URLs mentioned in an Asciidoc document should be verified (non 404) # features/support/checkLinks.feature:9
    Given An Asciidoc file                                                          # features/step_definitions/links.rb:1

FILE: README.adoc
[✓] http://jmbruel.github.io/teachingMaterials/
[✓] http://jmbruel.github.io/teachingMaterials
[✓] http://github.com/jmbruel/teachingMaterials
[✓] http://git-scm.com/
[✓] https://innopolis.university/en/
[✓] https://www.iut-blagnac.fr/
[✓] https://jmbruel.github.io/MobileModeling
[✓] http://mathsinfo.univ-tlse2.fr/accueil/formations/master-ice/
[✓] http://www.it-tallaght.ie/
[✓] http://www.it-tallaght.ie/index.cfm/page/course?id=128
[✓] https://api.codacy.com/project/badge/Grade/550a9c47c3d6426c9122765e45097a3c
[✓] https://app.codacy.com/gh/jmbruel/teachingMaterials?utm_source=github.com&utm_medium=referral&utm_content=jmbruel/teachingM
aterials&utm_campaign=Badge_Grade
[✓] https://sonarcloud.io/api/project_badges/measure?project=jmbruel_teachingMaterials&metric=sqale_index
[✓] https://img.shields.io/badge/Cucumber-Report-yellowgreen
[✓] https://reports.cucumber.io/reports/dfb48e1c-5d7e-46e0-8094-4cc06570bc7f
[✓] https://img.shields.io/badge/License-MIT-yellow.svg
[✓] https://opensource.org/licenses/MIT
[✓] https://github.com/jmbruel/teachingMaterials/blob/master/images/profiles.png
[✓] https://innopolis.university/
[✓] https://moodle.innopolis.university/course/view.php?id=724
[✓] https://jmbruel.github.io/InnopolisModernApplicationProduction/
[✓] https://www.univ-cotedazur.fr
[✓] https://lms.univ-cotedazur.fr/course/view.php?id=16573
[✓] https://www.iut-blagnac.fr
[✓] https://webetud.iut-blagnac.fr/course/view.php?id=352
[✓] http://iut-blagnac.github.io/POO/
[✓] https://wiki.eclipse.org/images/a/a2/Papyrus4education-logo.png
[✓] https://wiki.eclipse.org/Papyrus_for_Education
[✓] https://github.com/edu
[✓] https://www.freepik.com
[✓] https://www.flaticon.com/
[✓] http://creativecommons.org/licenses/by/3.0/
[✓] http://asciidoctor.org/
[✓] http://creativecommons.org/licenses/by-sa/3.0/

34 links checked.
```

Makefile

```
44      @echo "==> checking the URLs "
45      asciidoc-link-check $ -c $(EXCLUDE_URLS) $< > $@
46
47
48  $(CHECK_RES): checks/*.txt
49      @echo "======================================="
50      @echo "==> checking the fix "
51      @echo `date` > $(CHECK_RES)
52      cat checks/*.txt >> $(CHECK_RES)
53
54  deploy: check index.html
55      @echo "======================================="
56      @echo "==> Deploy updates "
57      rake && git commit -am "🤖 DEPLOY: last updates"; git pull; git push
58
59  clean:
60      rm *.html
```

Appendix A: Useful links

- Cucumber reference site
- Plein d'astuces pour les expressions
- A 10' tutorial
- Un très bon tutoriel en français
- An Android example
- An Angular example
- Clean Code