Introduction to BDD with Cucumber

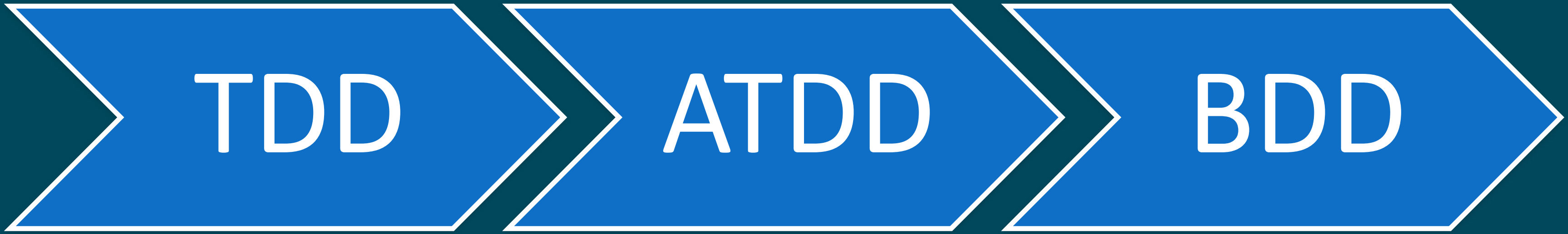
Nibu baby

XP
API
RSpec
DDD
Behat
Parking
Hudson
Kanban
Standup
Example
Impediment
Jenkins
ATDD
Pair
Technical
Programming
Planning
Huddle
Sprint
When
Burndown_Chart
Portfolio
JBehave
And
Story_Wall
Cucumber
Performance
WIP
DoD
Hip_Sprint
Jira
Background
Automation
BBDify
Concordion
Then
Webdriver
Specflow
BDD
Examples
Agile
ScenarioOutline
Scrum
Demo
Showcase
Scenario
Given
Specifications
SeleniumGrid
Retro
CI
TDD
Velocity
Testing
Gherkin
Epic
SQE

BDD?

BEHAVIOR DRIVEN DEVELOPMENT

An Evolution



TDD

ATDD

BDD

BDD

BDD is a second-generation, outside-in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology.

It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters.

BDD was developed by Dan North as a response to the issues encountered teaching TDD

Using examples to create a shared understanding and surface uncertainly to deliver software that matters

BDD

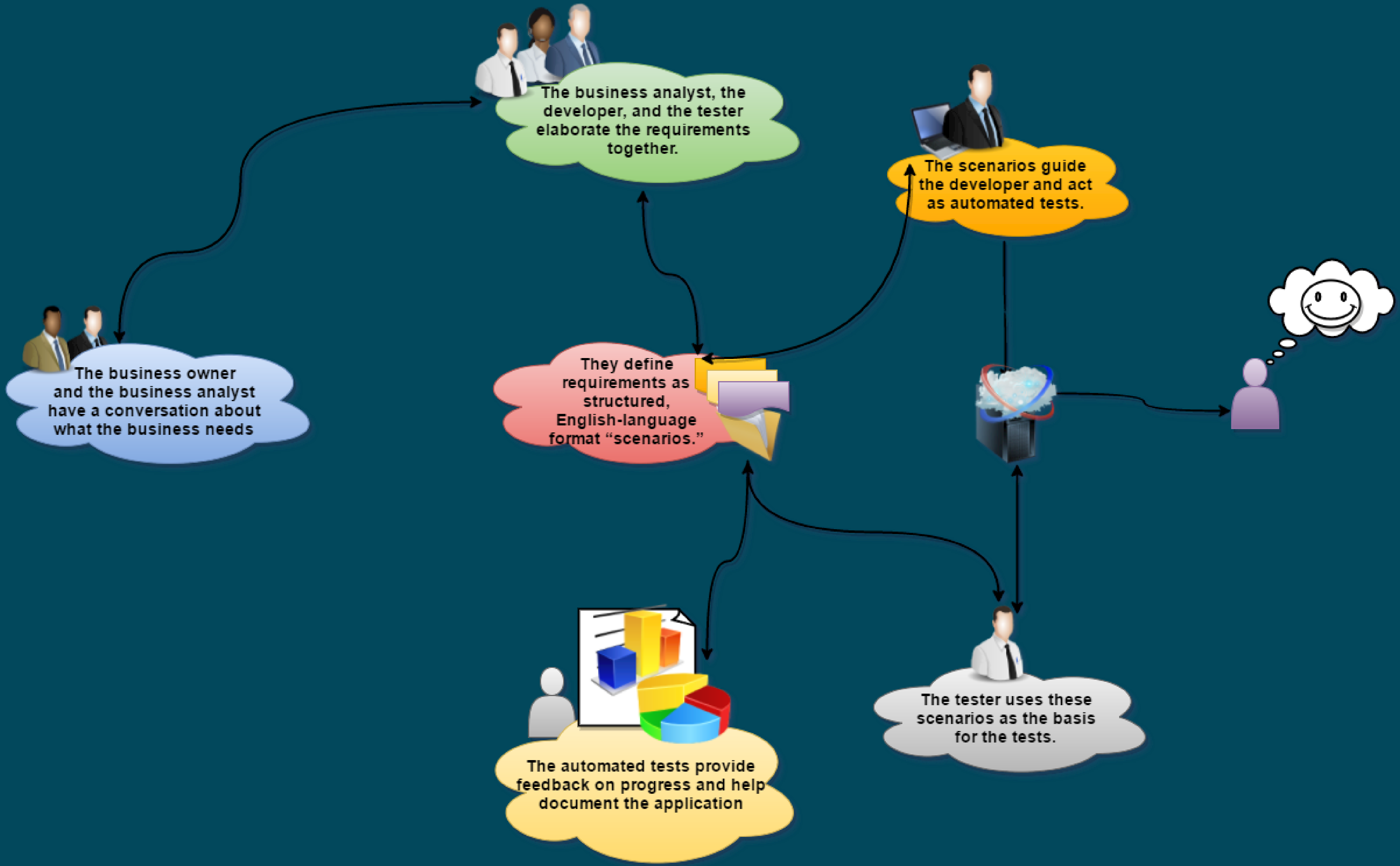
- Uses examples to illustrate behavior
- Core is around conversations and requirements discovery
- Uses conversations around examples, expressed In a form that can be easily automated, to reduce lost information and misunderstandings
- Starting with the business outcome, and working through high level functional areas to drill into specific stories with acceptance criteria.
- When writing examples, regression tests are a by-product

Examples

Given a context

When an event happens

Then an outcome should occur



The business owner and the business analyst have a conversation about what the business needs

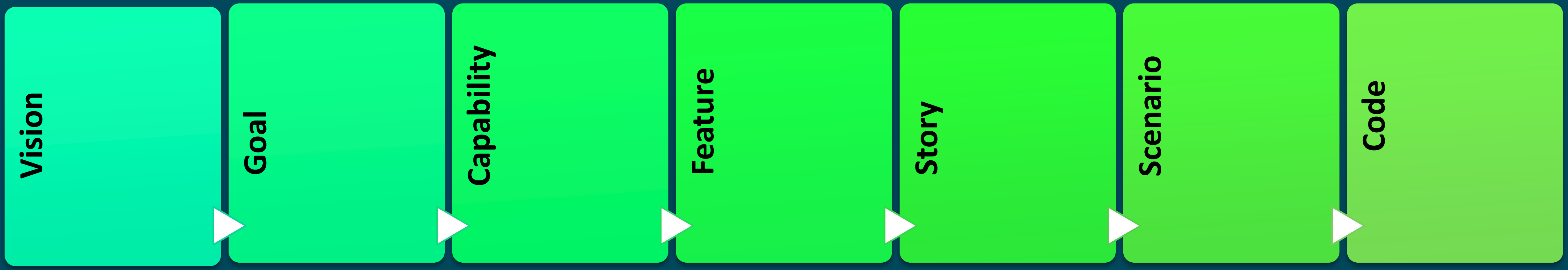
The business analyst, the developer, and the tester elaborate the requirements together.

They define requirements as structured, English-language format “scenarios.”

The scenarios guide the developer and act as automated tests.

The tester uses these scenarios as the basis for the tests.

The automated tests provide feedback on progress and help document the application



Vision

Goal

Capability

Feature

Story

Scenario

Code

Have enough conversation to get started. Find out where the riskiest bits are and where the system behaves unusually .

Chat, Discuss, Discover, Work out how and when , you are going to get feedback on the work you are about to do . Then and only then , reach for tools

[Liz Keogh]

Story

- What are we really trying to build ?
- It has to be a description of a requirement and its business benefit, and a set of criteria by which we all agree that it is “done” [Dann North]
- Written in business language
- Stories are the result of conversations between the project stakeholders, business analysts, testers and developers

Example

As a user
I want to publish a tweet
So that share information with my followers

Scenario

- Is set of conditions that story must meet for it to be accepted as complete
- Can be used to help discover the scope of the story or the feature
- Is not replacement for conversation
- Is the outcome of conversations with all players
- Scenario title should say what’s different
- The scenario should be described in terms of Givens, Events and Outcomes

Given [some context]

When [I do something]

Then [this happens]

BDD tools

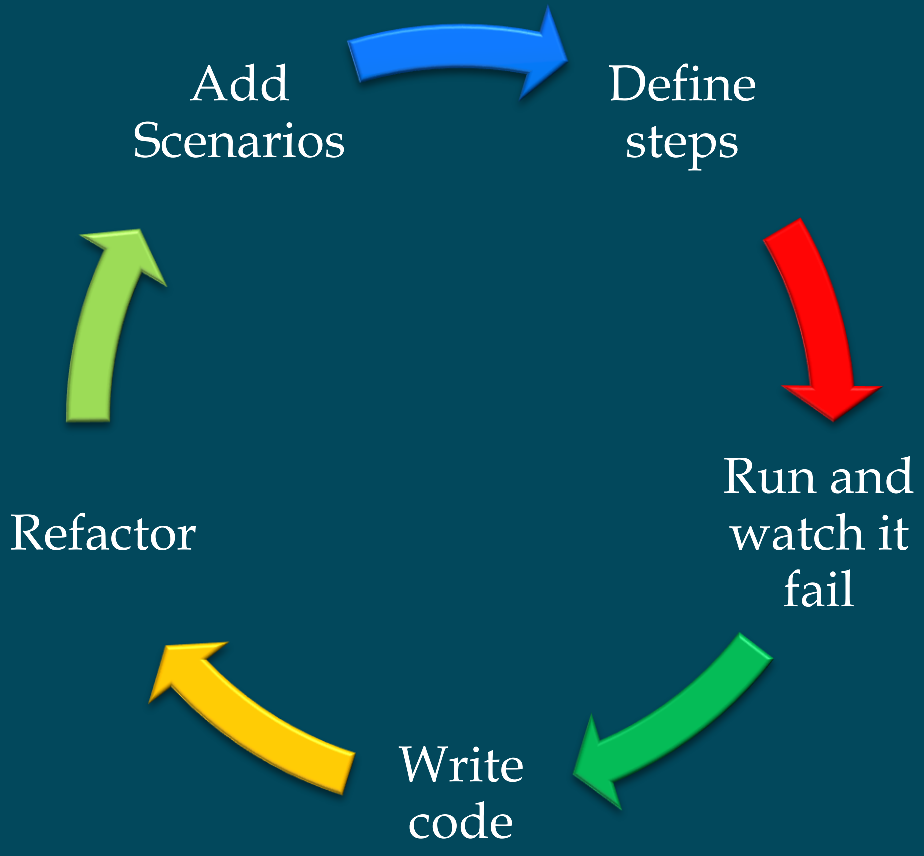
Few of them ….

Cucumber

Cucumber

- Collaboration tool
- ~~It's Testing tool~~
- Outside-In approach , where programmers incrementally write code and run the scenarios using Cucumber until the feature passes all the tests
- Supports about 60+ languages
- Helps to execute plain text functional descriptions as automated tests
- Automation can be written in own favorite language (Ruby, Java, c#.net , JavaScript, Scala, Groovy, Jython, Python, Perl, Go, C++ etc)
- writing scenarios before code enables programmers to be guided by an unambiguous specification

Life cycle



Add Scenarios

Define steps

Run and watch it fail

Write code

Refactor

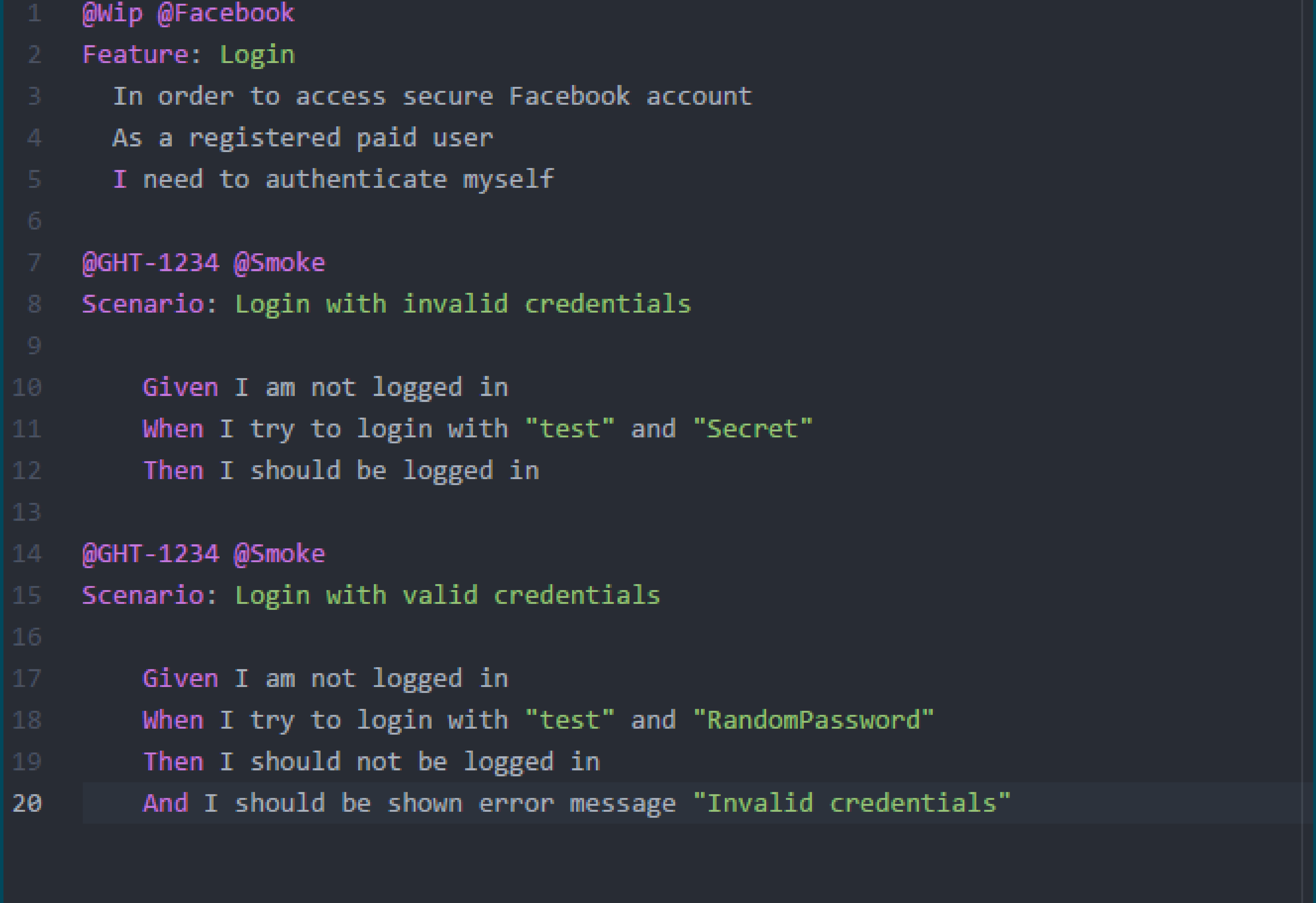
Gherkin

Means : “a small variety of cucumber, or a young green cucumber used for pickling.” [google]

Cucumber executes .feature files, and those files contain executable specifications written in a language called Gherkin.

Gherkin supports 60+ languages.

Gherkin is designed to be easy to learn by non-programmers, yet structured enough to allow concise description of examples to illustrate business rules in most real-world domains. [www. cucumber.io]



```
@Wip @Facebook
Feature: Login
  In order to access secure Facebook account
  As a registered paid user
  I need to authenticate myself

@GHT-1234 @Smoke
Scenario: Login with invalid credentials

    Given I am not logged in
    When I try to login with "test" and "Secret"
    Then I should be logged in

@GHT-1234 @Smoke
Scenario: Login with valid credentials

    Given I am not logged in
    When I try to login with "test" and "RandomPassword"
    Then I should not be logged in
    And I should be shown error message "Invalid credentials"
```

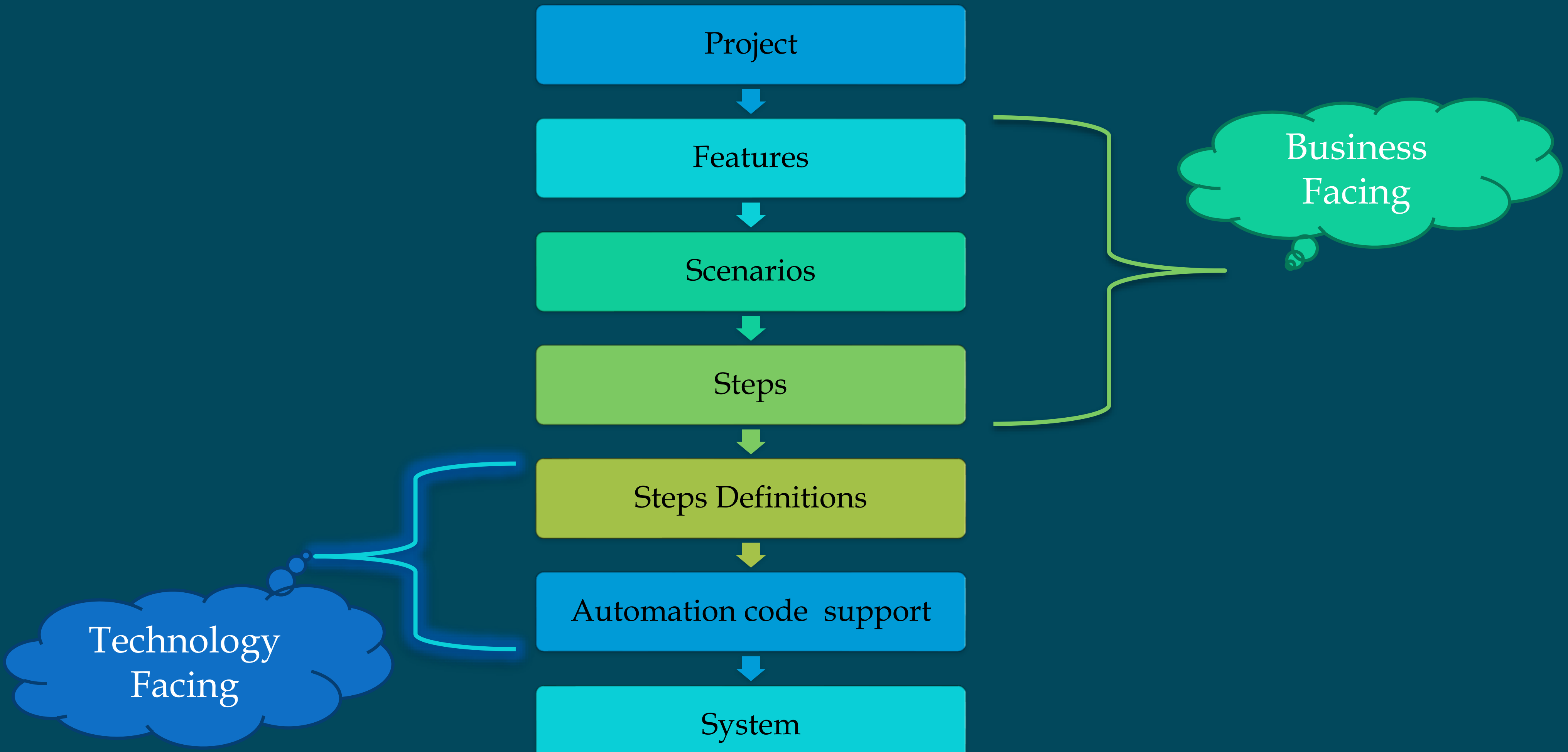
Gherkin Keywords

- Feature
- Scenario
- Given, When, Then, And, But
- Background
- Scenario Outline
- Examples

More helpful keywords

- """ (Doc Strings)
- | (Data Tables)
- @ (Tags)
- # (Comments)

Cucumber testing stack



Let’s explore

Benefits

- Better communication
- Ubiquitous language (Same/being present everywhere)
- Better collaboration between stakeholders, BA, QA and Developers
- Formal documentation is not required
- Living Documentation (Single source of truth)
- Quick feedback (If the automated tests are included in the CI)
- Right Focus, Right Product, Product Right

Disadvantages

- It requires high level of business engagement and collaborations
- May not work well in silos
- Poorly written tests can lead to higher test-maintenance costs

References

- http://dannorth.net
- http://lizkeogh.com
- http://behaviour-driven.org
- https://cucumber.io

- BDD in action [John Ferguson]
- The cucumber book [Matt wynne and Aslak Hellesoy]