Getting Started with TDD/BDD: The Quality Process Does Not Improvise

Katy García Bedoya. *Tech Lead Test Automation Engineer*
Deiny Johana Betancourt. *QC Analyst*

TAE week - 2018

Globant
we are ready

Schedule

- Introducing Behavior-Driven Development
- A Gherkin primer
- BDD — Best Practices
- Key benefits of BDD / Disadvantages and potential challenges of BDD
- Demo

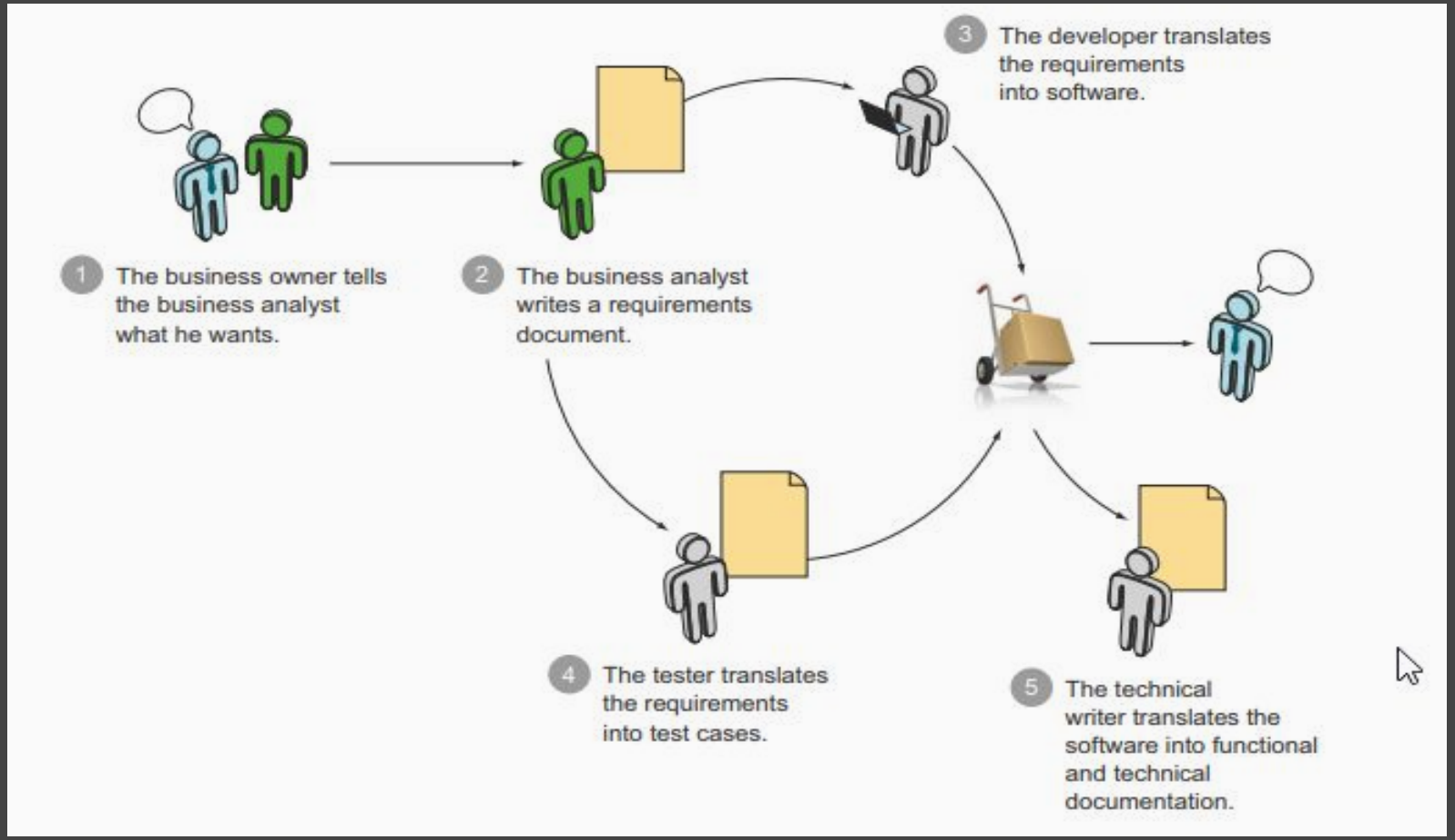


Figure 1.1 The traditional development process provides many opportunities for misunderstandings and miscommunication.

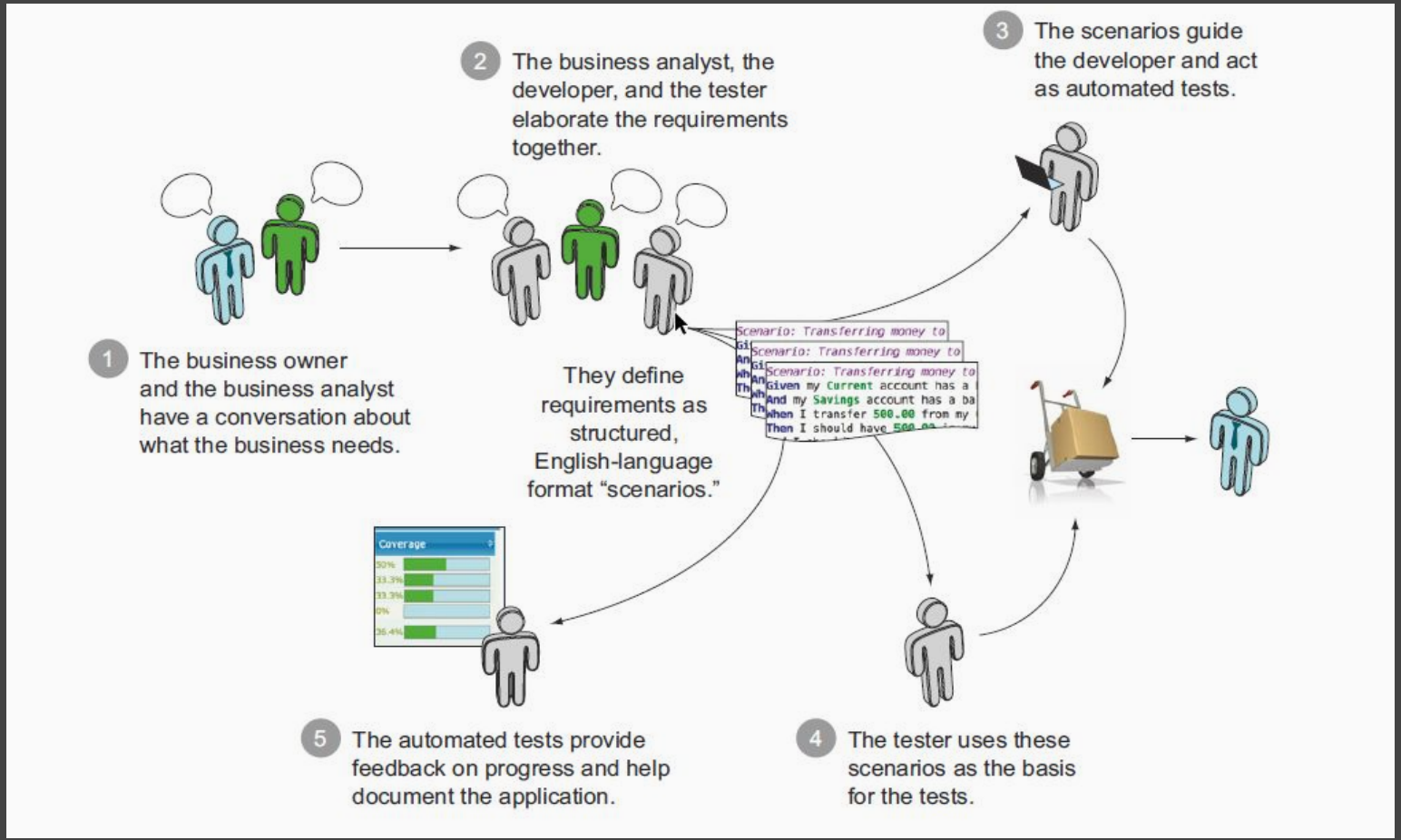
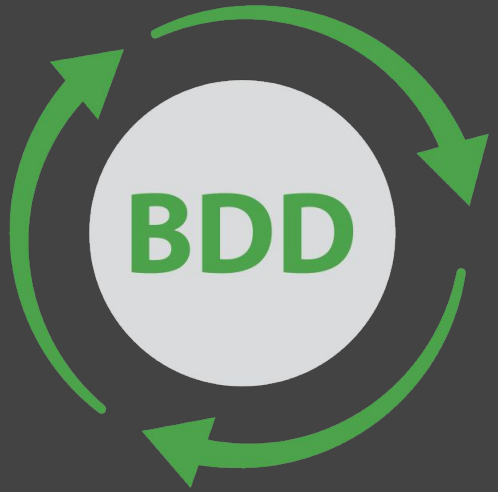


Figure 1.2. BDD uses conversations around examples, expressed in a form that can be easily automated, to reduce lost information and misunderstandings.

Introducing Behavior-Driven Development

Behavior-Driven Development (BDD) is a set of software engineering practices designed to help teams build and deliver more valuable, higher quality software faster.

But most importantly, BDD provides a common language based on simple, structured sentences expressed in English (or in the native language of the stakeholders) that facilitate communication between project team members and business stakeholders.



BDD was originally designed as an improved version of TDD

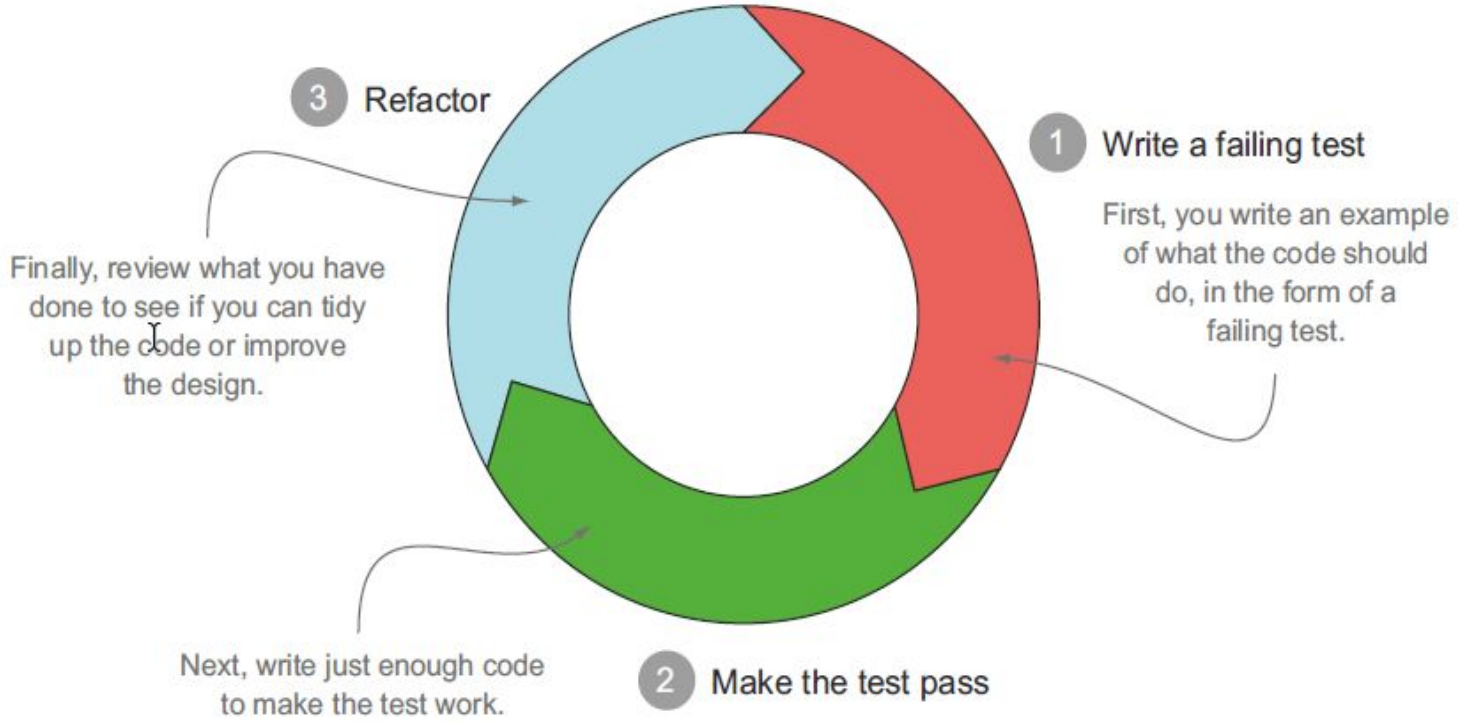


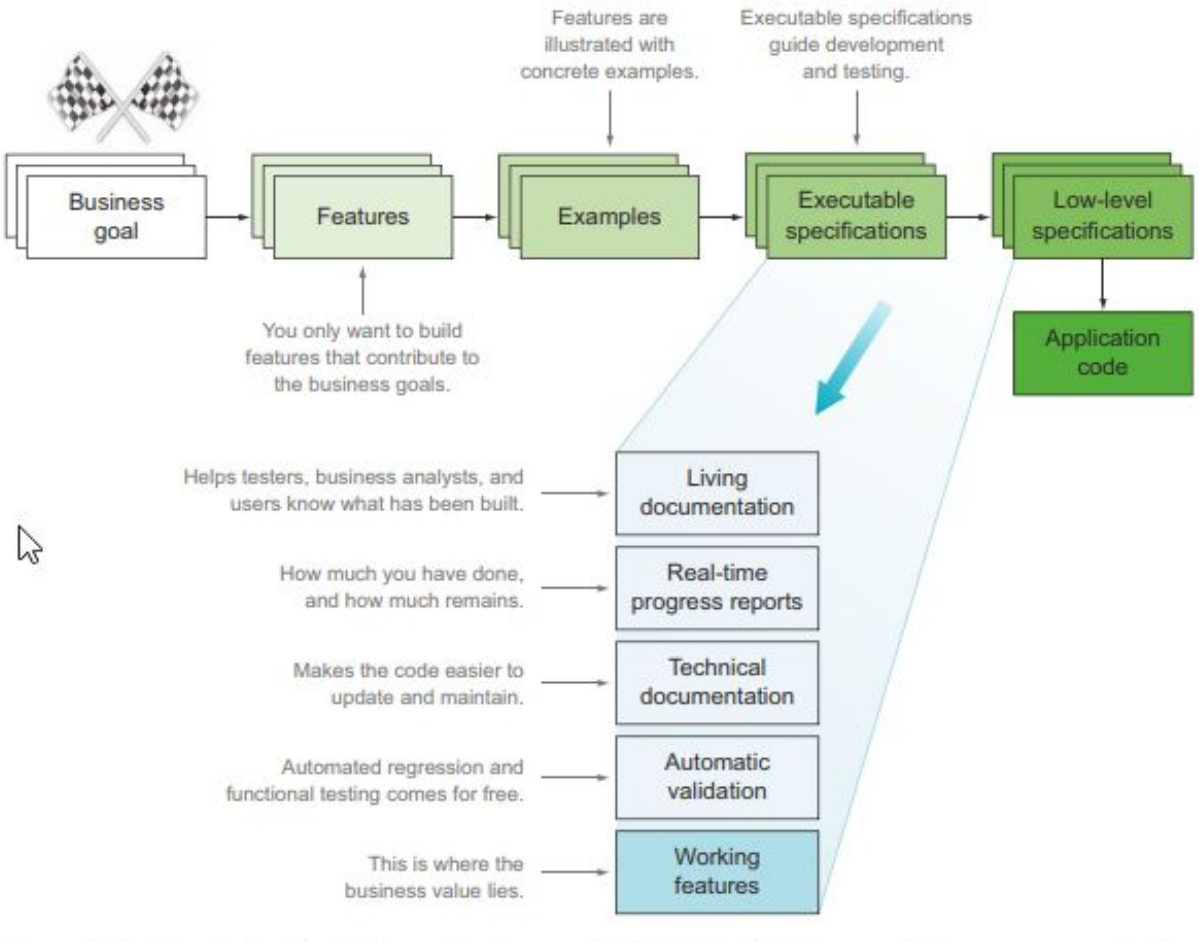
Figure 1.3. Test-Driven Development relles on a simple, three-phase cycle.

Introducing Behavior-Driven Development

```
public class
BankAccountTest {
@Test
public void testTransfer()
{...}
@Test
public void testDeposit()
{...}
}
```

```
public class WhenTransferringInternationalFunds
{
@Test
public void
should_transfer_funds_to_a_local_account()
{...}

@Test
public void
should_transfer_funds_to_a_different_bank()
{...}
...
@Test
public void
should_deduct_fees_as_a_separate_transaction()
{...}
...
}
```



High-level and low-level executable specifications are typically implemented using different tool sets.

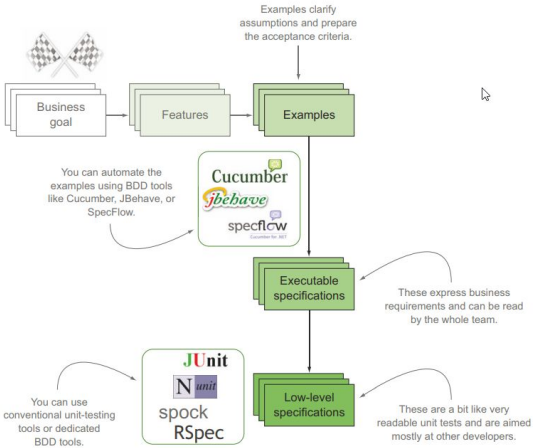


Figure 1.4. The principal activities and outcomes of BDD.

A Gherkin primer

```
Scenario Outline: Earning interest

Given I have an account of type <account-type> with a balance of <initial-balance>

When the monthly interest is calculated

Then I should have earned at an annual interest rate of <interest-rate>

And I should have a new balance of <new-balance>

Examples:
```

initial-balance	account-type	interest-rate	new-balance
10000	current	1	10008.33
10000	savings	3	10025
10000	supersaver	5	10041.67

BDD — Best Practices

Points to take care for “Feature files”:

- Avoid long descriptions
- Chose a single format all across your features
- Focus on features that deliver business value
- Work together to specify features

Points to take care for “Scenarios and steps”:

- Think of Scenario Or Scenario Outline
- Keep scenarios short by hiding implementation details
- Use Given-When-Then in the right order
- Make Scenarios Atomic
- Cover both the happy and non-happy paths
- Use declarative steps rather than imperative

Good Example (Declarative):

```
Scenario: Login
Given I have user credentials
When Sign in
Then I should see the user is signed in
```

As you can see, using declarative steps in the scenario makes it shorter and more readable.

Bad Example (Imperative):

```
Scenario: Login
Given I am on the login page
When I fill “username” with “ABC”
And I fill password with “XYZ”
And I checked the “Remember Me” checkbox
And I click on the “Submit” button
Then I should log into the system
And I should see then ‘Welcome’ message
```

Points to take care for “Tags”:

How to decide name of the tag

Use only relevant tags when there is a necessity else managing tags will become add-on task and overhead.).

Frequency of Execution: @daily, @hourly, @nightly

Dependencies: @database, @fixtures, @local, @proxy

Progress: @wip, @todo, @implemented, @blocked

Level: @functional, @acceptance, @smoke, @sanity

Environment: @integration, @test, @stage, @live

Tag your feature smartly

```
For example, you could tag a feature with the story number in a
bug tracking system like JIRA.
Ex:
@Jira-story #PROJ-33
```

Don’t tag a scenario with the same tag used for tagging feature

If you tag an entire feature then any scenarios within the feature automatically inherit this tag. Don’t use too many unnecessary tags in the project, as this will overcomplicated your feature files.

Key benefits of BDD

- Reduced waste
- Reduced costs
- Easier and safer changes (Living documentation)
- Faster releases

Disadvantages and potential challenges of BDD

- BDD requires high business engagement and collaboration
- BDD works best in an Agile or iterative context
- Poorly written tests can lead to higher test-maintenance costs

Globant
we are ready


Globant
we are ready