Successfully Implementing BDD in an Agile World

SMARTBEAR

BDD Overview

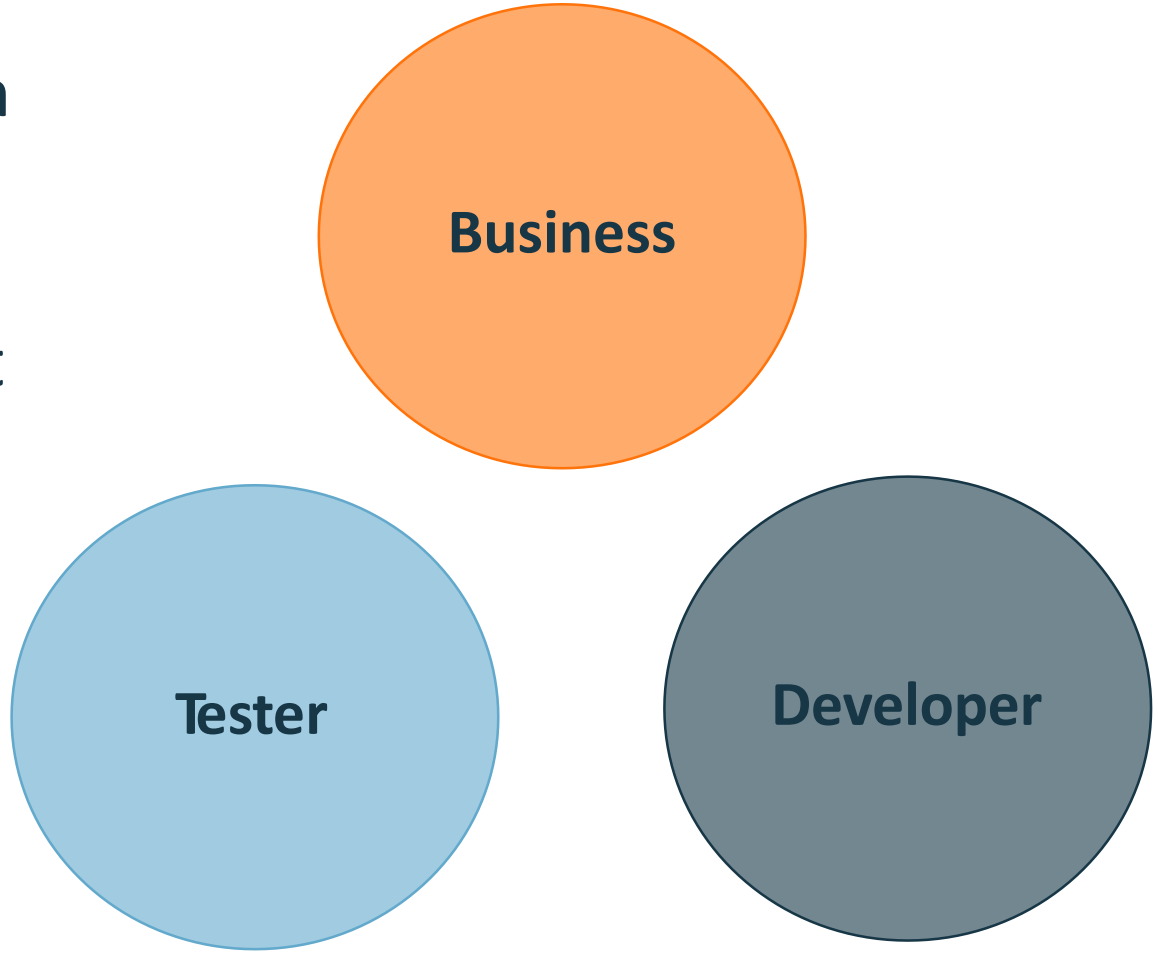
Behavior Driven Development (BDD):

*Is a software development process where teams create simple steps on how an application **should** behave from a **user's** perspective.*

What is BDD?

- Discovery, Formulation, and Automation
- Create a **shared understanding** based on examples
- Use the **examples** to drive development
- Create and share **living documentation**

The 3 Amigos



What are the problems BDD helps address?

1. Business needs are not well understood by the people who want the software
2. Business needs are not well understood by people who deliver and test the software
3. The software is low quality with lots of bugs
4. The cost of changes over time increases rapidly

Making the Move to BDD

Opportunities and Challenges

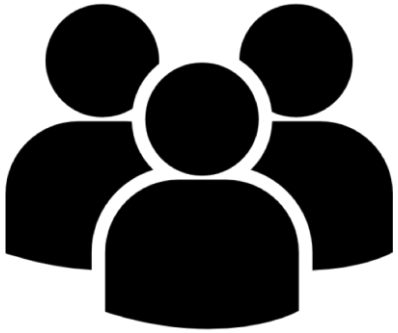
Challenges:

1. Misalignment between various stakeholders on your team (product, dev, QA)
2. Producing products that don't address your user's needs
3. No universal language between departments
4. Pressure to release faster without sacrificing quality

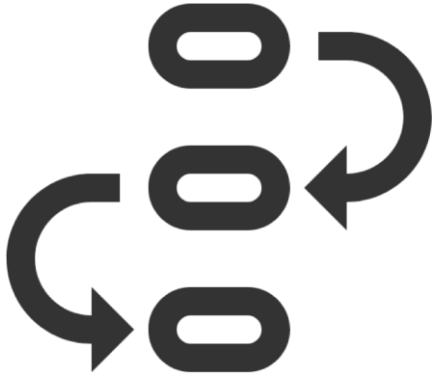
Benefits:

1. Better communication and collaboration
2. Deliver value more frequently
3. Find and fix defects early and often
4. Minimize waste
5. Avoid development of features nobody wants

In order to succeed you need to take a holistic approach



People



Processes

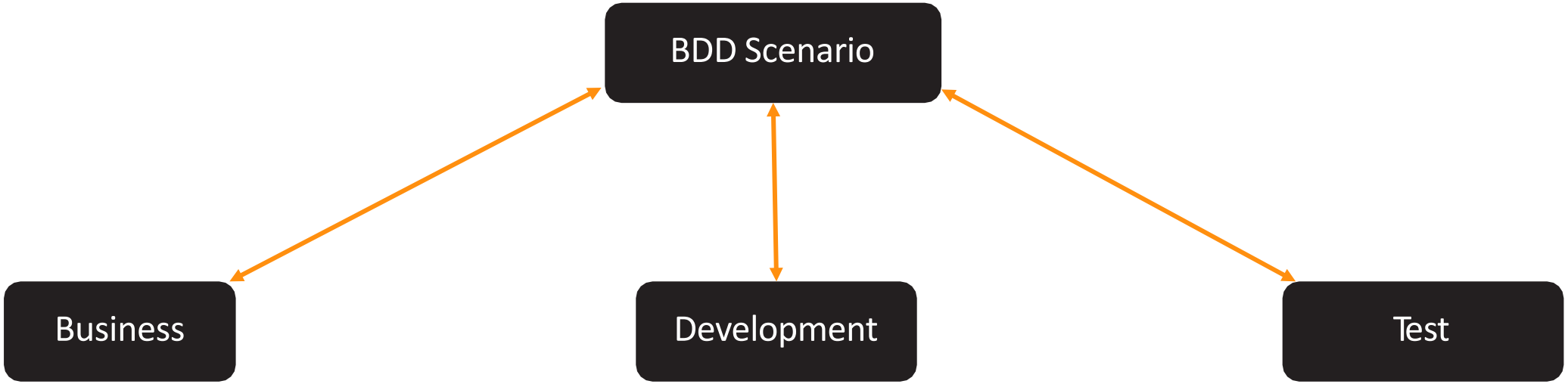
Tools

Let’s start with people…

… it’s all about **collaboration**

Roles

Using a common Language



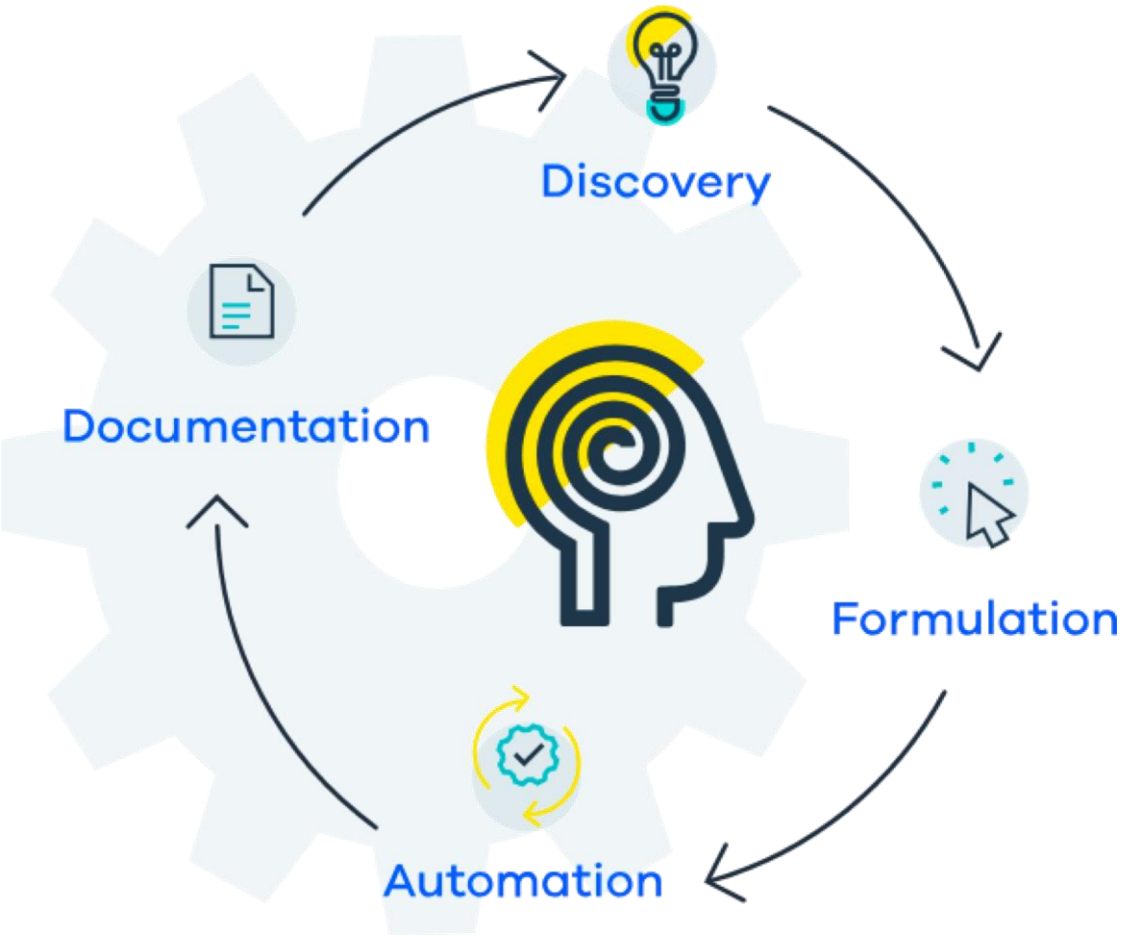
- Defines requirements in spoken-word scenarios
- Obtains feedback in clear, example heavy, and easy to understand language

- Making the Scenario “pass” or “work” guides development efforts and provides focus
- Scenario is easily translated directly into automated tests
- Given / When / Then structure provides simple ways to ask for clarification

- Scenarios contain Test definitions (Action, expected result)
- Structure allows for easy reporting and traceability

Process

The BDD Process



Deliberate Discovery

Goal: Try and learn as much as possible before writing any code

Have conversations about user stories and acceptance criteria using **concrete examples**

Who should be involved in Deliberate Discovery?

- Product Owners
- Business Analysts
- Domain Experts
- Users
- Developers
- UX Designers
- Testers

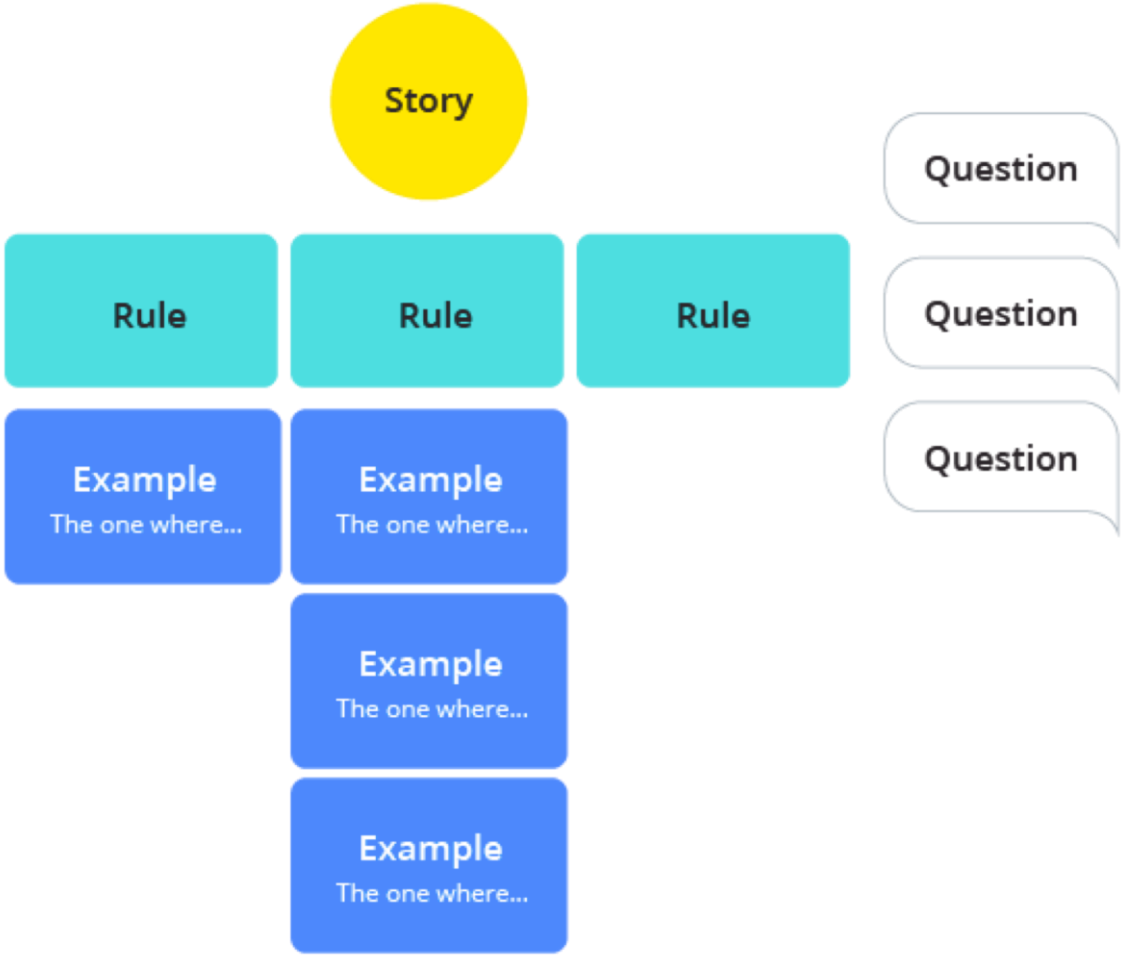
Running a Discovery Workshop

From Matt Wynne:

- Three Amigos Meetings
- Keep them short! (25 minutes/user story)
- Use Example Mapping
  - User Story—user stories that were developed in deliberate discovery process
  - Rules—acceptance criteria for the user story containing agreed-upon constraints
  - Examples—concrete examples covering each rule (may have more than one per rule)
  - Questions—unknowns that arise when exploring rules, examples, or assumptions made to move forward

Example Mapping—Session

- Set-Up:
  - Use a pack of four-color index cards
  - Remove distractions
- Running a Session:
  1. Write the story in yellow
  2. Write acceptance criteria, or “rule” in blue
  3. Write concrete examples on green
  4. Capture questions on red
  5. Vote to see if the user story is ready for development



Record the Results

- Use Gherkin Syntax
- Create Executable Specification without code

Test steps

View tests

Raw version

```
1  Given the account balance is  $100
2    and the machine contains enough money
3    and the card is valid
4  When the Account Holder requests  = amount
5  Then the ATM should dispense  = amount
6    and the account balance should be  = ending_balance
7    and the card should be returned
```

Add step

Step 1: Focus on the Value First

- What are the benefits of your feature?

- First step: start with answering “why”
  - Why should we develop this feature?
  - Why will this feature help my user?
- A feature should have a description that provides context to a need being met
  - In order to [get a benefit]
  - As a [role]
  - I want [a feature]
- Example: HipTest & Slack
  - Everyone on team needs a notification?
  - Only HipTest users should be able to get notified

Step 1: Focus on the Value First

- What are the benefits of your feature?

- First step: start with answering "why"
  - Why should we develop this feature?
  - Why will this feature help my user?
- A feature should have a description that provides context to a need being met
  - In order to [get a benefit]
  - As a [role]
  - I want [a feature]
- Example: HipTest & Slack
  - Everyone on team needs a notification?
  - Only HipTest users should be able to get notified

Step 2: Be Declarative When Writing Scenarios

- Focus on the what?
- Two styles of writing your test scenario
  - Imperative
    - Long scenarios with low-level steps describing how to navigate the user interface
  - Declarative
    - Gherkin syntax
      - Given is your setup
      - When is your action
      - Then is your assertion
    - Text describes the what not the how

Step 2: Be Declarative When Writing Scenarios

- The Declarative Style of Writing—An Example

- Given I am logged in
- When I add an item to the cart
- And I click on the cart
- Then I should see the item I added

Why is this Declarative?

- It doesn't matter how someone did the login→ username, password, touchID
- What does matter: someone logged in

Declarative Example for HipTest & Slack

Given I have activated slack in the settings of my HipTest project
When I query /hiptest TestRunID from slack
Then I should see the breakdown of tests by statuses for testRunID

Step 3: Automate (if you'd like) Your BDD Scenarios

- Build a test automation framework
  - Focus on building a framework that will scale
- Your first execution will fail--> the feature is not yet implemented
- Implement your feature
- Run the automated BDD scenarios to show the feature is completed
- Repeat