Behaviour Driven Development : theGardener roots

BDD : theGardener roots

- Introduction
- Full BDD example on a library
- Improve the process with theGardener
- Conclusion

Introduction

Goal : make you want to try or try the BDD again

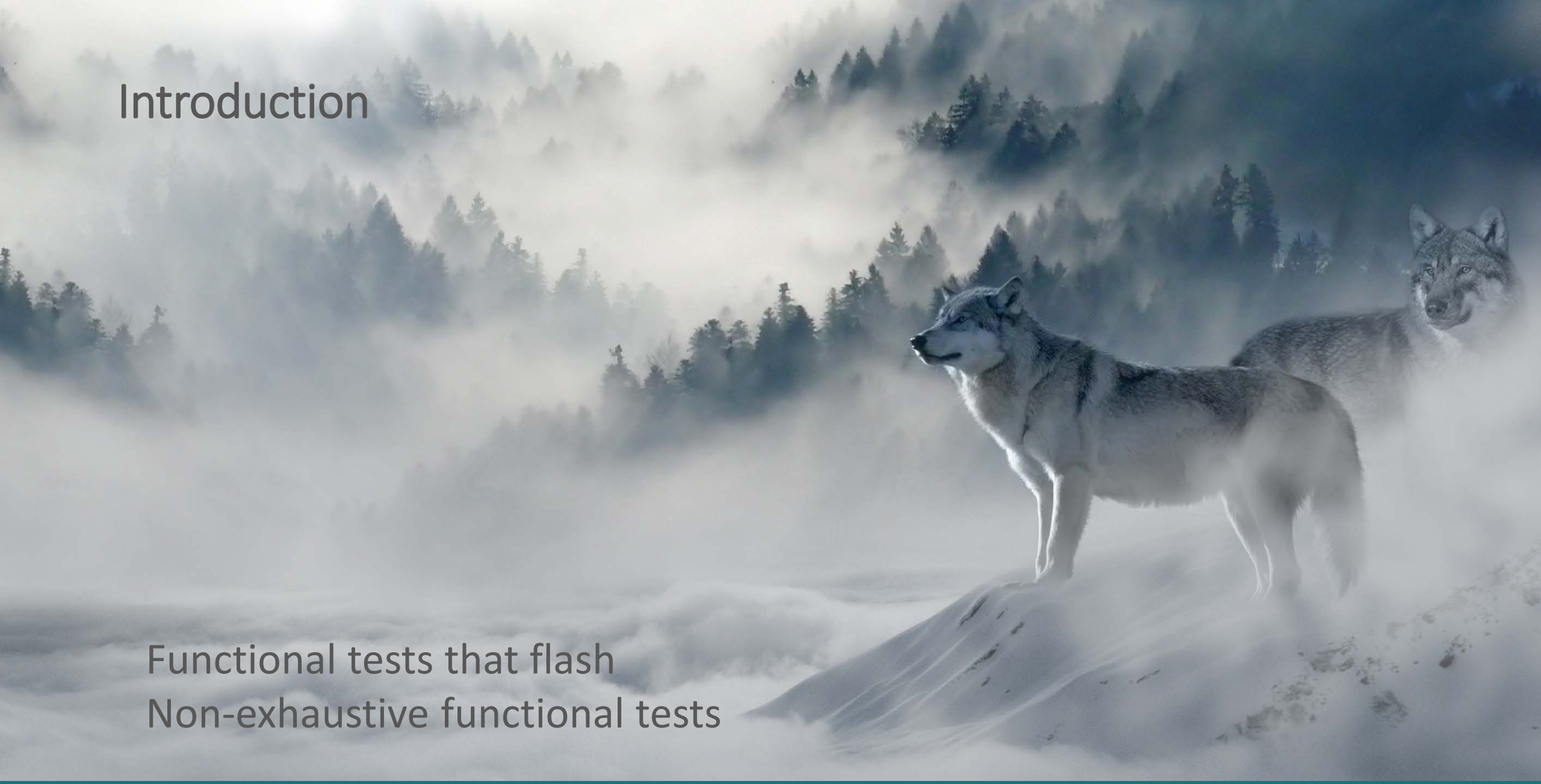
Introduction

A product owner during the demo of the product after an iteration

Introduction

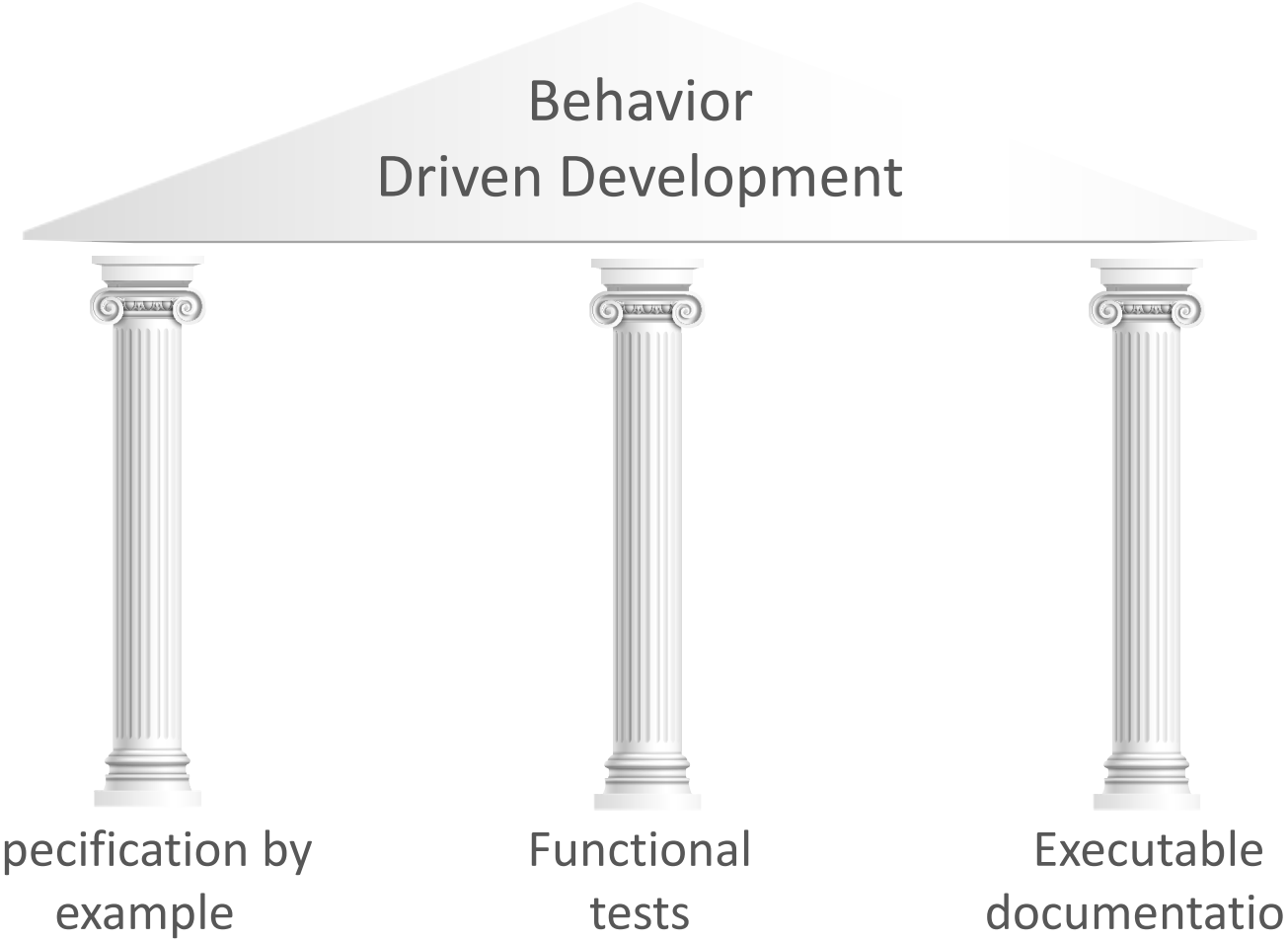
Functional documentation after many iterations and readjustments of the need

Introduction



Functional tests that flash
Non-exhaustive functional tests

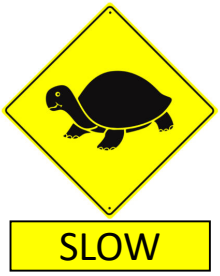
Introduction



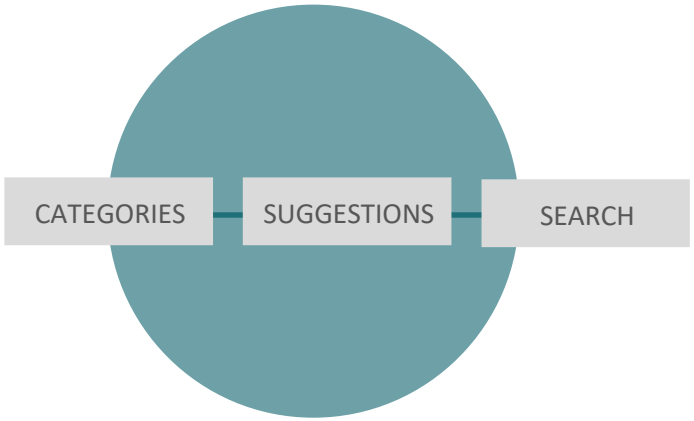
Introduction | Specification by an example

```
Scenario: suggested suggestions are popular, available and adapted to the age of the user

    Given the user "Tim"
    and he is "4" years old
    and the popular categories for this age are
      | categoryId  | name              |
      | cat1        | Walt Disney       |
      | cat2        | Bedtime stories   |
    and the available books for those categories are
      | bookId | bookTitle                | categoryId |
      | lv11   | Peter Pan                | cat1       |
      | lv21   | The tortoise and the hare | cat2       |
    When we ask for "2" suggestions
    Then the suggestions are
      | bookId   | bookTitle                  | categoryId |
      | lv11     | Peter Pan                  | cat1       |
      | lv21     | The tortoise and the hare  | cat2       |
```

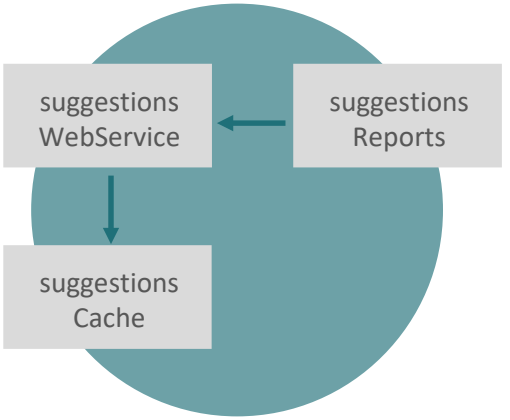

Introduction | Functional tests



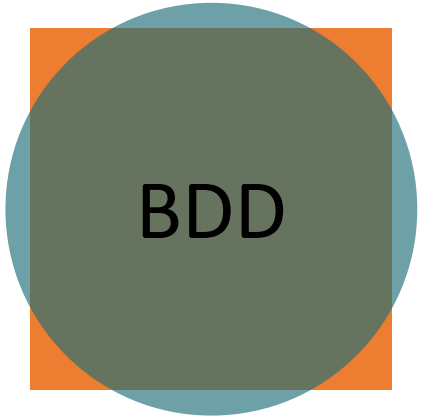
HARD



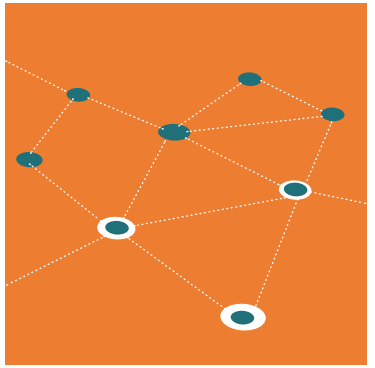
Between systems



On a system between components



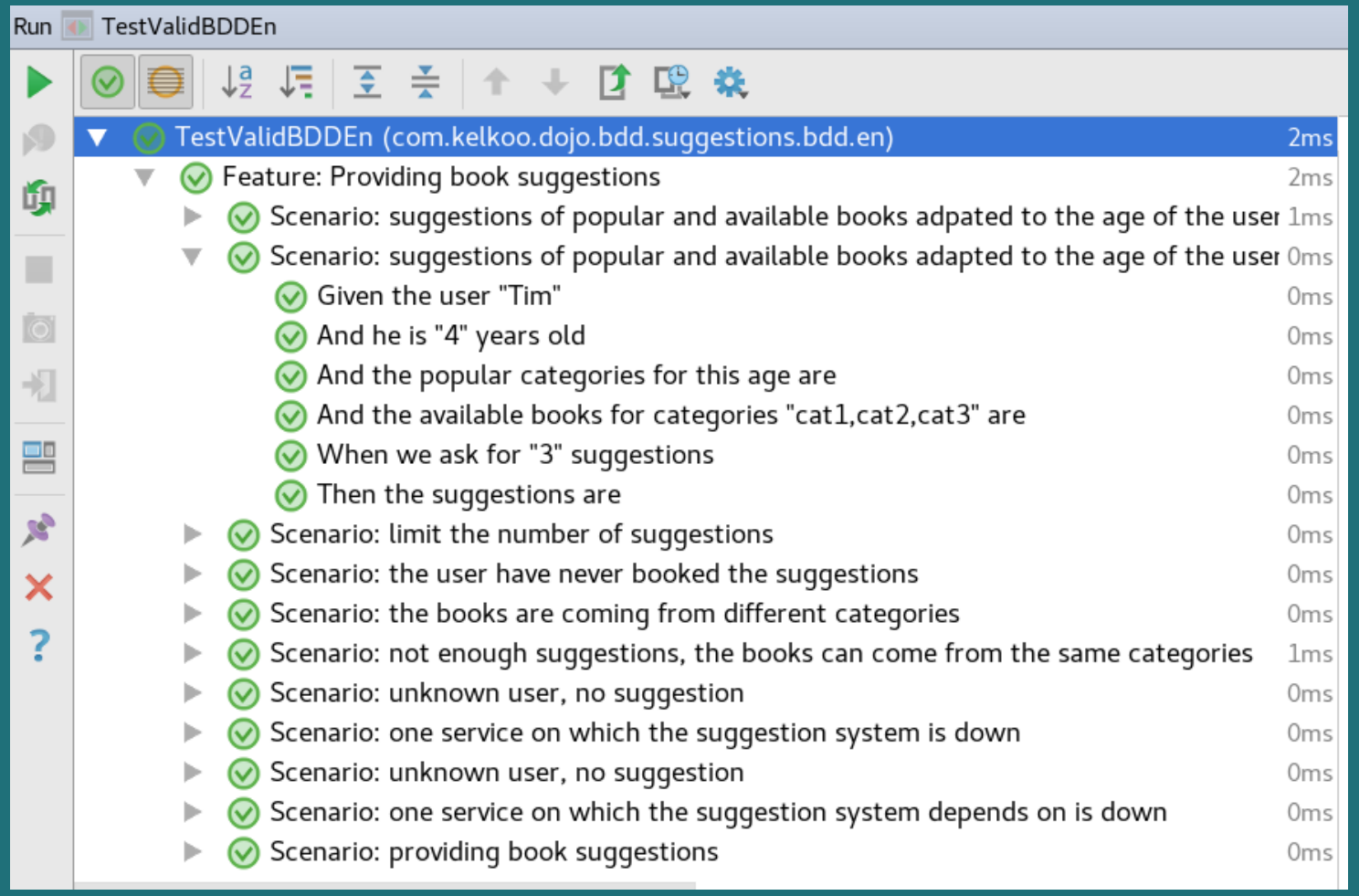
On a component



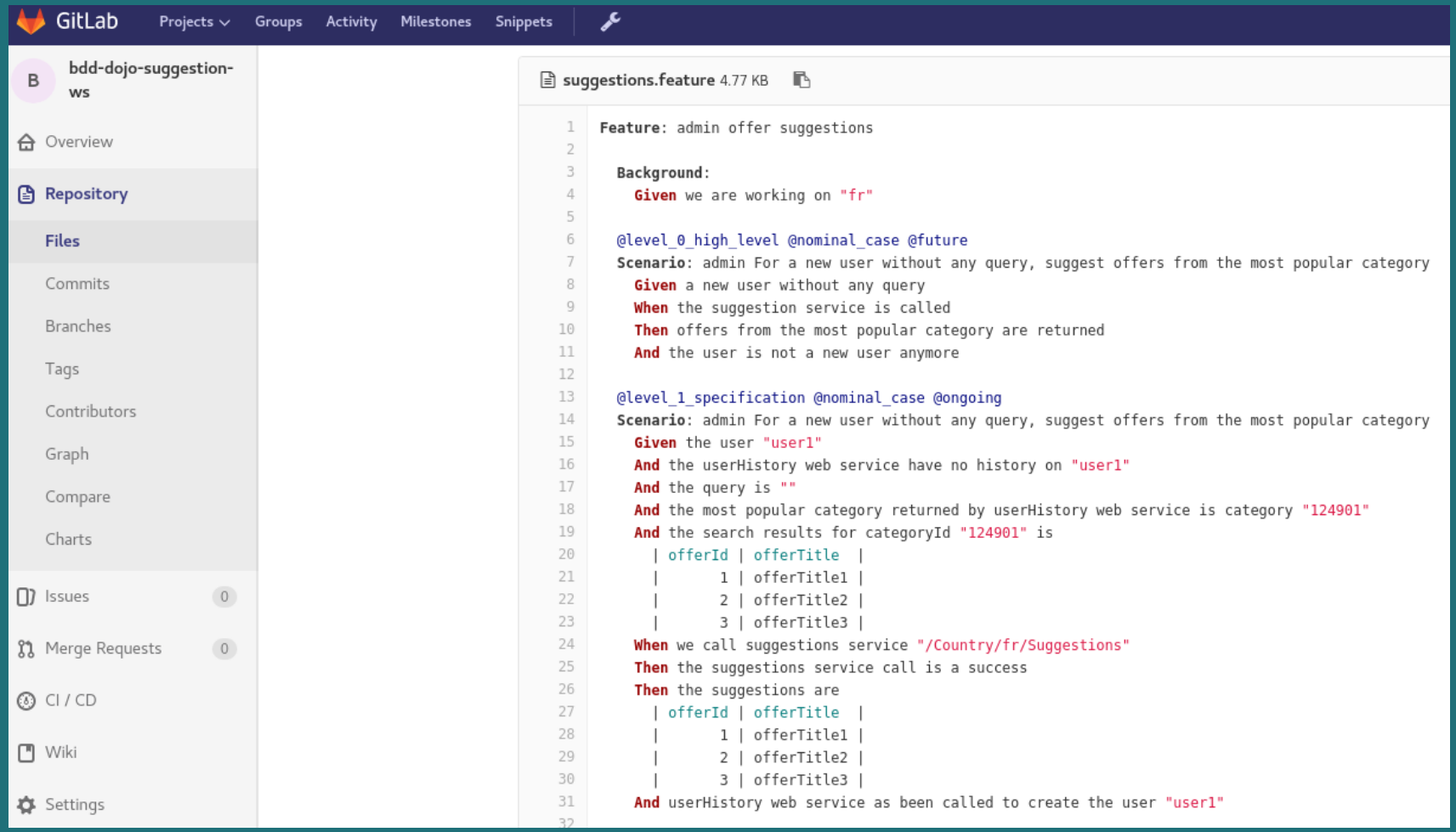
On a class

EASY

Introduction | Functional tests



Introduction | Executable Documentation



Library | User Story to be implemented

PO

DEV

Library | User Story to be implemented

CATEGORIES

Categories of books,
popular categories by age

User

USERS

Users, ages,
books already red...

SUGGESTIONS

Provides book
suggestions

SEARCH

Provides books, textual search,
multi-criteria search
(category, popularity, availability ...)

BOOKING

Booking service,
Available books

Library | User Story to be implemented

As a user of the library,
I wish to book suggestions
to make discoveries

Acceptance criteria :

- Book not read by the user
- Book available

Library | User Story to be implemented

PO

User Story

As a user of the library, I wish to book suggestions to make discoveries

Suggestions must be appropriate to the age of the user

For a better discovery, the books must come from different categories

Library | User Story to be implemented

DEV

User Story

As a user of the library, I wish to book suggestions to make discoveries

Focus on how to recover books, forgets that the book must be unread by the user

The simplest way : research the popularity of books

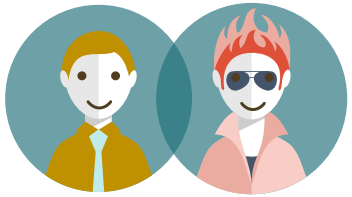
Library | Write scenarios in collaboration

PO

```
Scenario: provide book suggestions

   Given a user
   When  we ask for suggestions
   Then  the suggestions are popular and
         available books adapted to the age of the user
```

Missing example!

Library | Write scenarios in collaboration



PO DEV

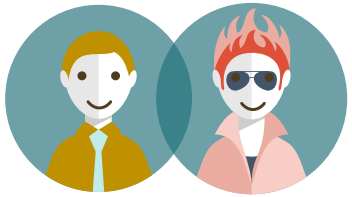
```
Scenario: provide book suggestions

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
      | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat2,cat3" are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
   When we ask for "3" suggestions
   Then the suggestions are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
```

Missing limit number of suggestions

Missing : never read

Missing : different categories

Library | Write scenarios in collaboration



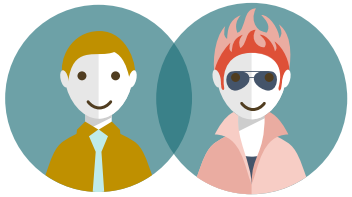

```
Scenario: provide book suggestions

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
      | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat2,cat3" are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
   When we ask for "3" suggestions
   Then the suggestions are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
```

Missing : never read

Missing : different categories

limit number of suggestions

Library | Write scenarios in collaboration



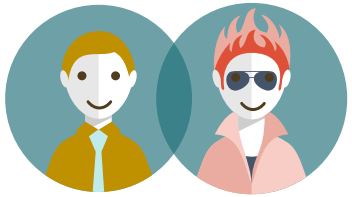
PO DEV

```
Scenario: provide book suggestions

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
      | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat2,cat3" are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
   When we ask for "3" suggestions
   Then the suggestions are
      | bookId  | title                       | categoryId  |
      | lv11    | Peter Pan                   | cat1        |
      | lv21    | Picture book about farm     | cat2        |
      | lv31    | The tortoise and the hare   | cat3        |
```

Missing : never read

different categories

Library | Write scenarios in collaboration



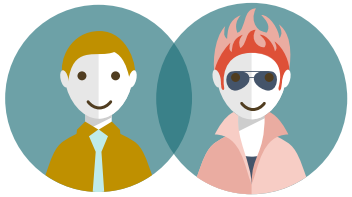
PO DEV

```
Scenario: provide book suggestions

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
      | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat2,cat3" are
      | bookId  | title                        | categoryId  |
      | lv11    | Peter Pan                    | cat1        |
      | lv21    | Picture book about farm      | cat2        |
      | lv31    | The tortoise and the hare    | cat3        |
   When we ask for "3" suggestions
   Then the suggestions are
      | bookId  | title                        | categoryId  |
      | lv11    | Peter Pan                    | cat1        |
      | lv21    | Picture book about farm      | cat2        |
      | lv31    | The tortoise and the hare    | cat3        |
```

What are we testing ?
Prefer several scenario

never read

Library | Write scenarios in collaboration

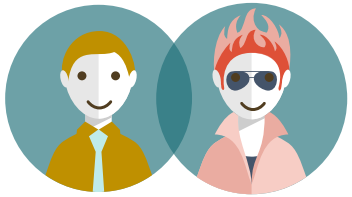


PO DEV

```
Scenario: suggested suggestions are popular, available
          and adapted to the age of the user

    Given the user "Tim"
    And he is "4" years old
    And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
    And the available books for categories "cat1,cat2" are
      | bookId  | title                     | categoryId  |
      | lv11    | Peter Pan                 | cat1        |
      | lv21    | Picture book about farm   | cat2        |
    When we ask for "2" suggestions
    Then the suggestions are
      | bookId  | title                     | categoryId  |
      | lv11    | Peter Pan                 | cat1        |
      | lv21    | Picture book about farm   | cat2        |
```

Scenario 1 : nominal case
=> minimal

Library | Write scenarios in collaboration



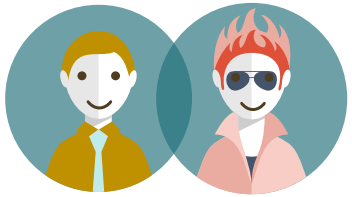
PO DEV

```
Scenario: limit number of suggestions

    Given the user "Tim"
    And he is "4" years old
    And the popular categories for this age are
      | categoryId  | name             |
      | cat1        | Walt Disney      |
      | cat2        | Picture books    |
    And the available books for categories "cat1,cat2" are
      | bookId  | title                      | categoryId  |
      | lv11    | Peter Pan                  | cat1        |
      | lv21    | Picture book about farm    | cat2        |
    When we ask for "1" suggestions
    Then the suggestions are
      | bookId  | title                      | categoryId  |
      | lv11    | Peter Pan                  | cat1        |
```

Scenario 2 : nominal case

Simplify it again !

Library | Write scenarios in collaboration

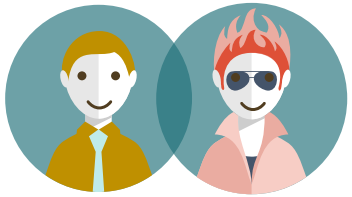


PO DEV

```
Scenario: limit number of suggestions

    Given a user
    And "3" books are available in popular categories
            adapted to his age

    When we ask for "2" suggestions
    Then "2" suggestions are proposed
             among the previous books
```

Scenario 2 : nominal case

Library | Write scenarios in collaboration



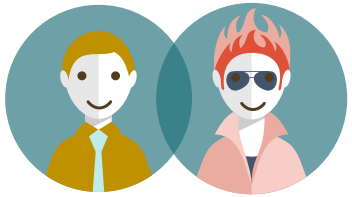
PO DEV

```
Scenario: the user has never red the books that are suggested

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
     | categoryId  | name             |
     | cat1        | Walt Disney      |
     | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat3" are
     | bookId  | title                      | categoryId  |
     | lv11    | Peter Pan                  | cat1        |
     | lv31    | The tortoise and the hare  | cat3        |
   And the user has already booked the following books
     | bookId  | title                      | categoryId  |
     | lv11    | Peter Pan                  | cat1        |
   When we ask for "1" suggestions
   Then the suggestions are
     | bookId  | title                      | categoryId  |
     |         |                            |             |
     | lv31    | The tortoise and the hare  | cat3        |
```

Scenario 3 : nominal case

Roll out the algorithm

Library | Write scenarios in collaboration



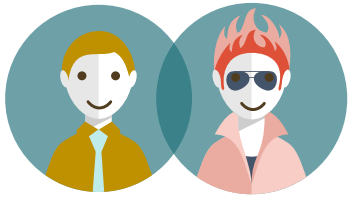
PO DEV

```
Scenario: suggested books come from different categories

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
     | categoryId  | name             |
     | cat1        | Walt Disney      |
     | cat3        | Bedtime stories  |
   And the available books for categories "cat1,cat3"  are
     | bookId  | title                      | categoryId  |
     | lv11    | Peter Pan                  | cat1        |
     | lv12    | Pinocchio                  | cat1        |
     | lv31    | The tortoise and the hare  | cat3        |
   When we ask for "2" suggestions
   Then the suggestions are
     | bookId  | title                      | categoryId  |
     | lv11    | Peter Pan                  | cat1        |
     |         |                            |             |
     | lv31    | The tortoise and the hare  | cat3        |
```

Scenario 4 : nominal case

Roll out the algorithm

Library | Write scenarios in collaboration

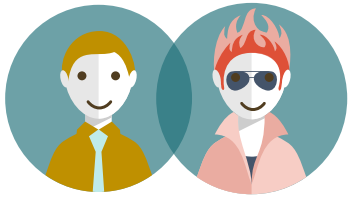



```
Scenario: if there is not enough suggestions,
          we can propose books from the same categories

   Given the user "Tim"
   And he is "4" years old
   And the popular categories for this age are
     | categoryId  | name             |
     | cat1        | Walt Disney      |
   And the available books for categories "cat1,cat3 are
     | bookId  | title                    | categoryId  |
     | lv11    | Peter Pan                | cat1        |
     | lv12    | Pinocchio                | cat1        |
   When we ask for "2" suggestions
   Then the suggestions are
     | bookId  | title                    | categoryId  |
     | lv11    | Peter Pan                | cat1        |
     | lv12    | Pinocchio                | cat1        |
```

Scenario 5 : limit case

Roll out the algorithm

Library | Write scenarios in collaboration

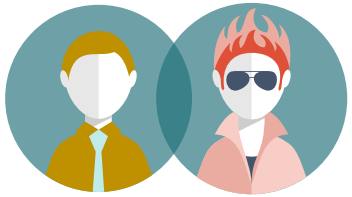


PO DEV

```
Scenario: unknown user, no suggestion

    Given the user "Lise"
    And the user is unknown
    When we ask for "3" suggestions
    Then there is non suggestions
```

Scenario 6 : limit case

Library | Write scenarios in collaboration

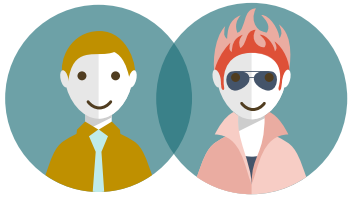


PO DEV

```
Scenario: one service on which the suggestion system
          depends on is down

   Given the use "Tim"
   And impossible to get information on the user
   When we ask for "3" suggestions
   Then the system is temporary not available
```

Scenario 7 : error case

Library | Write scenarios in collaboration

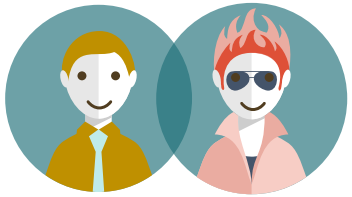


PO DEV

```
 Scenario: suggested suggestions are popular, available
           and adapted to the age of the user

Given the user from http://my.library.com/user/Tim
   | field  | value |
   | userId | Tim   |
   | age    | 4     |
And the categories from http://my.library.com/category?popular=true&age=4
   | categoryId | categoryName    |
   | cat1       | Walt Disney     |
   | cat2       | Picture books   |
   | cat3       | Bedtime stories |
And the books from http://my.library.com/search?categories=cat1,cat2,cat3&available=true
   | bookId | bookTitle                | categoryId |
   | b11    | Peter Pan                | cat1       |
   | b21    | Picture book about farm  | cat2       |
   | b31    | The tortoise and the hare | cat3       |
And the books from http://my.library.com/user/Tim/books
   | bookId | bookTitle | categoryId |
   | b11    | Peter Pan | cat1       |
When we call http://localhost:9998/suggestions?userId=Tim&maxResults=3
Then the http code is "200"
Then the suggestions are
   | bookId | bookTitle                | categoryId |
   | b21    | Picture book about farm  | cat2       |
   | b31    | The tortoise and the hare | cat3       |
```

Scenario 1 technichal version

Library | Organize scenarios



PO DEV

As a user of the library,
I wish to book suggestions
to make discoveries

Scenario 1

Scenario 2

Scenario 7

Scenario 0

Scenario 5

Scenario 3

Scenario 4

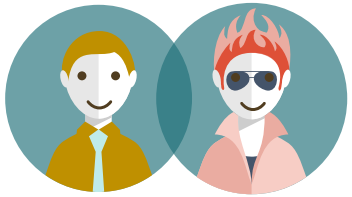
Scenario 6

Scenario 7

Scenario 1

Scenario 6

Library | Organize scenarios



PO DEV

As a user of the library,
I wish to book suggestions
to make discoveries

	@nominal_case	@limit_case	@error_case
@level_0_ high_level	Scenario 0		
@level_1_ specification	Scenario 1 Scenario 2 Scenario 3 Scenario 4	Scenario 5 Scenario 6	Scenario 7
@level_2_ technical	Scenario 1	Scenario 6	Scenario 7

Library | Without specification by example

Suggestions should be linked to the user !

Demo

User Story

Acceptance Criteria

PO

Implementation

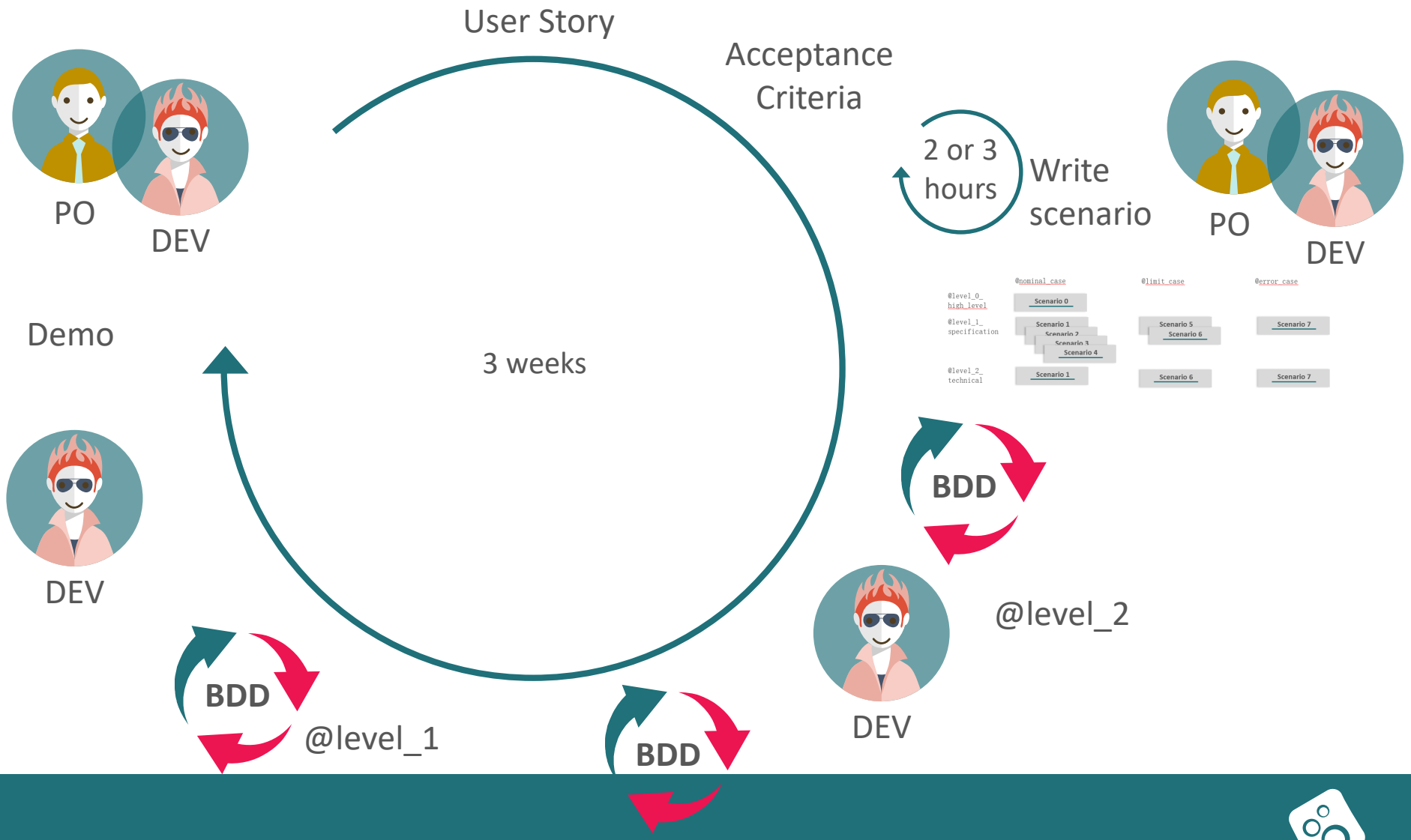
DEV

Implementation

Implementation

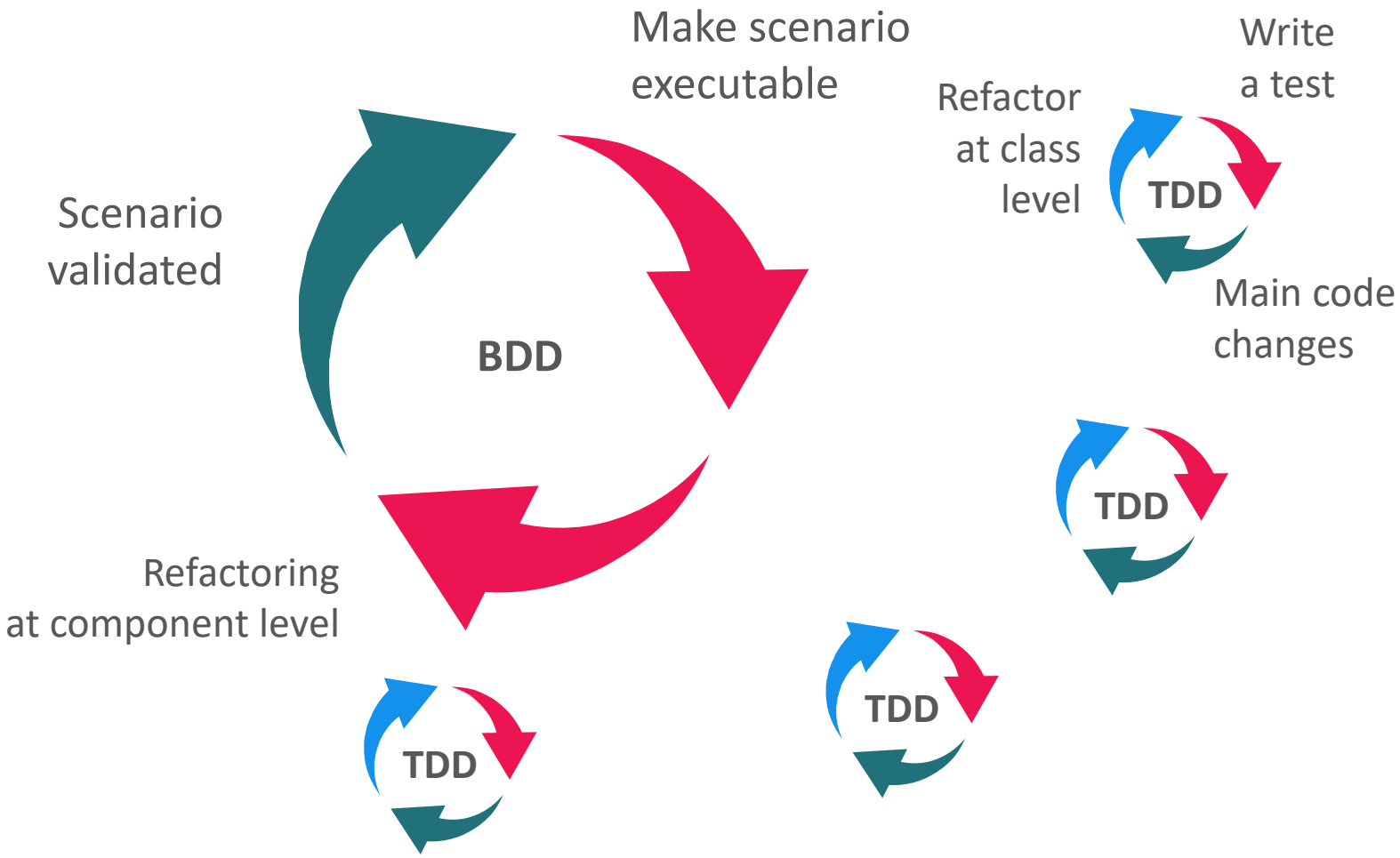
DEV

Library | Sprint roadmap



Library | Development cycles

DEV



Library | Make scenario executable

DEV

User
Name

Categories

Users

Booking

Search

Suggestions

Suggestions

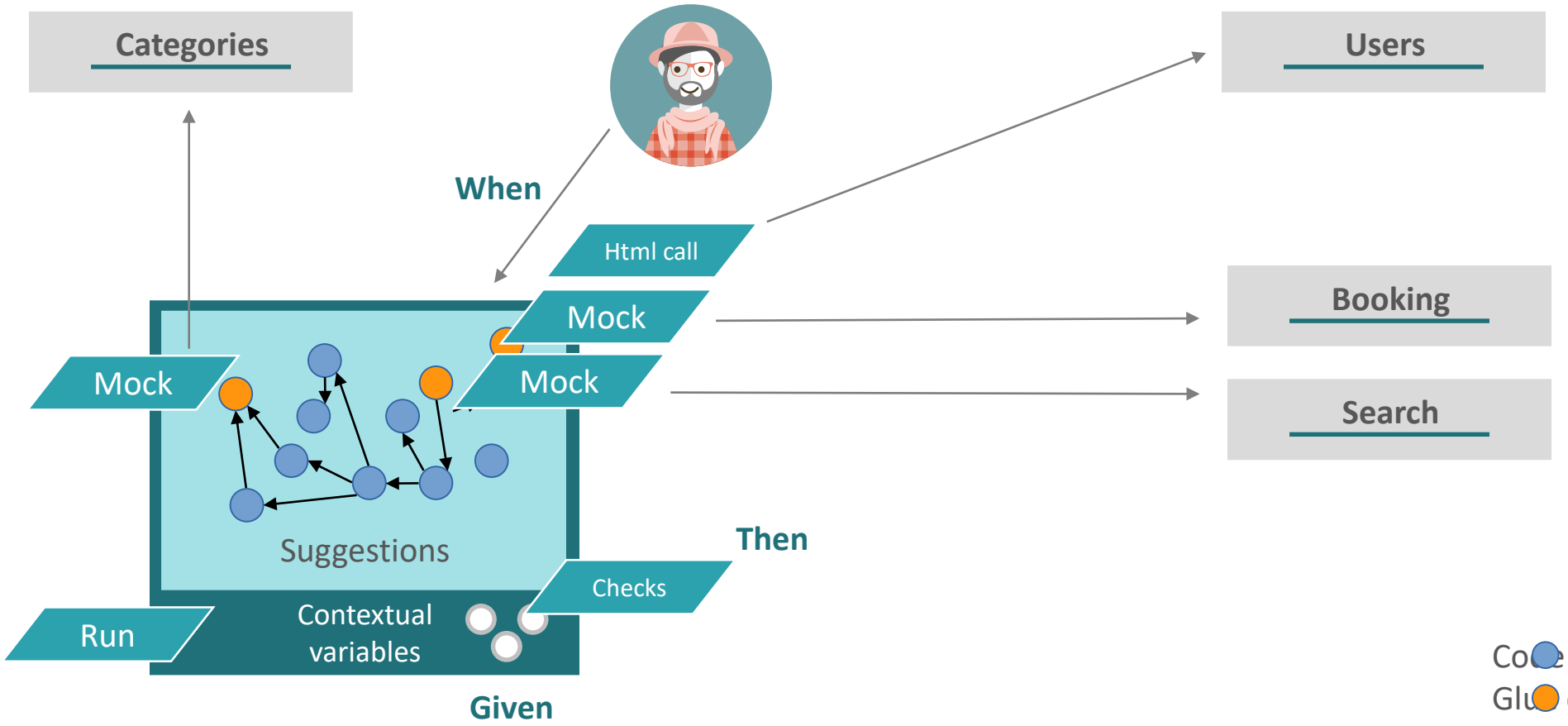
Popular
categories
by age

Available books
by category

Available books by category

Library | Make scenario runnable

DEV

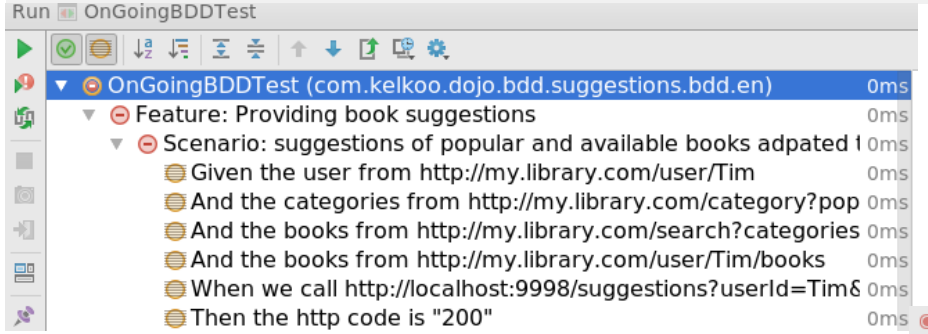


Code de production
Glue code for BDD

Library | Make scenario runnable

DEV

```
@level_2_technical_details @nominal_case @ongoing
Scenario: suggestions of popular and available books adapted to the age of the user, he have never booked suggestions

    Given the user from  http://my.library.com/user/Tim
      | field       | value            |
      | userId      | Tim              |
      | age         | 4                |
    And the categories from http://mt.library.com/category?popular=true&age=4
      | categoryId  | categoryName     |
      | cat1        | Walt Disney      |
      | cat2        | Picture book     |
```



The dev is guided


```
You can implement missing steps with the snippets below:

@Given("^the user from http://my.library.com/user/Tim$")
public void the_user_from_http_my_library_com_user_Tim(DataTable arg1) throws Throwable {
    // Express the Regexp above with the code you wish you had
    // For automatic conversion, change DataTable to List<YourType>
    throw new PendingException();
}
```

Library | Make scenario runnable

DEV

Glue code between steps and main code

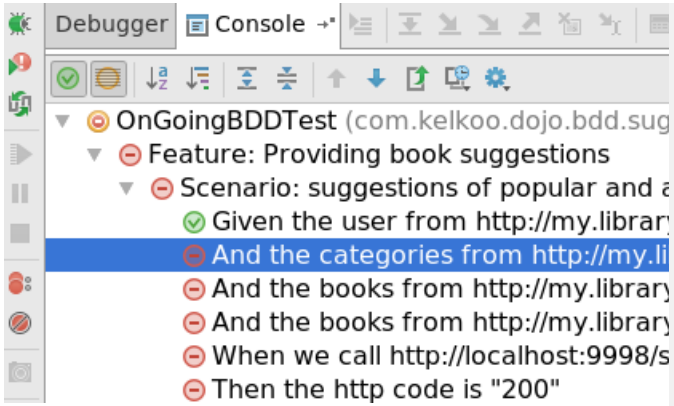
```
@Given( “^the user from http://my.library.com/user/([^\”]*)$” )
public void given_the_user_from_ws(String userId, List<FieldValue> values) throws Throwable {
     FieldValues fieldsValues = new FieldValues(values);
     user.setUserId(userId);
     user.setAge(fieldsValues.getAsInteger(field:” age” ));
     when(usersWSClientMock.retrieveUser(user.getUserId())).thenReturn(user);
}
```

Contextual variable

Library | Make scenario runnable

BDD

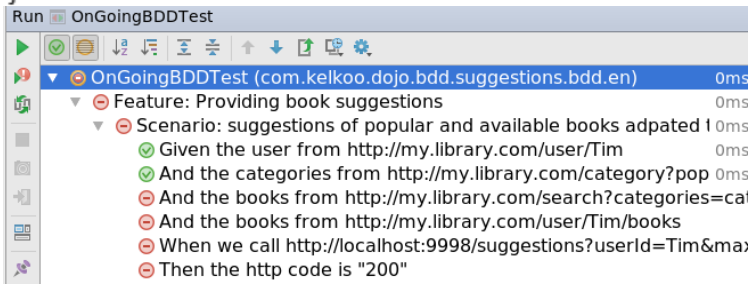
DEV



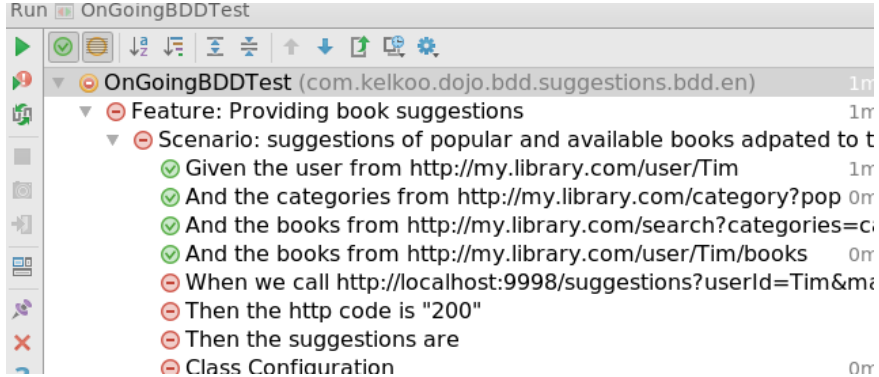

```
@level_2_technical_details @nominal_case @ongoing
Scenario: suggestions of popular and available books adapted to the age of the user,
          he have never booked suggestions
    Given the user from  http://my.library.com/user/Tim
      | field       | value            |
      | userId      | Tim              |
      | age         | 4                |
    And the categories from http://mt.library.com/category?popular=true&age=4
      | categoryId  | categoryName     |
      | cat1        | Walt Disney      |
      | cat2        | Picture book     |
      | cat3        | Bedtime stories  |
    And the books from http://mt.library.com/categories=cat1,cat2,cat3&available=true
      | bookId      | bookTitle        | categoryId       |
      | b11         | Walt Disney      | cat1             |
```

Define mocks behavior

```
@Given("^the categories from http://my.library.com/category\\?popular=([^\"]*)&age=(\\d+)$")
public void given_the_categories_from_categories_ws(Boolean popular , Integer age, List<Category> popularCategoriesGivenAgeUser)
    when(categoriesWSClientMock.retrieveCategories( popular, user.getAge())).thenReturn(popularCategoriesGivenAgeUser);
}
```

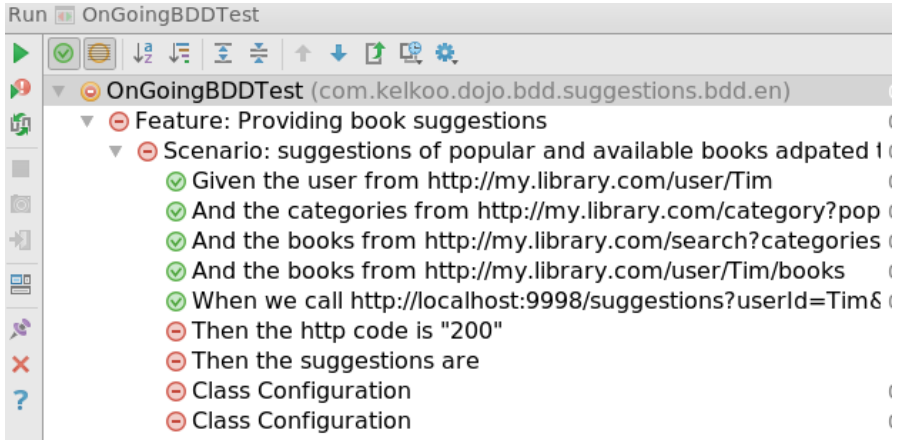


Library | Make scenario runnable

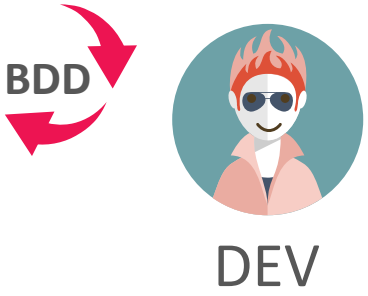


Main code call

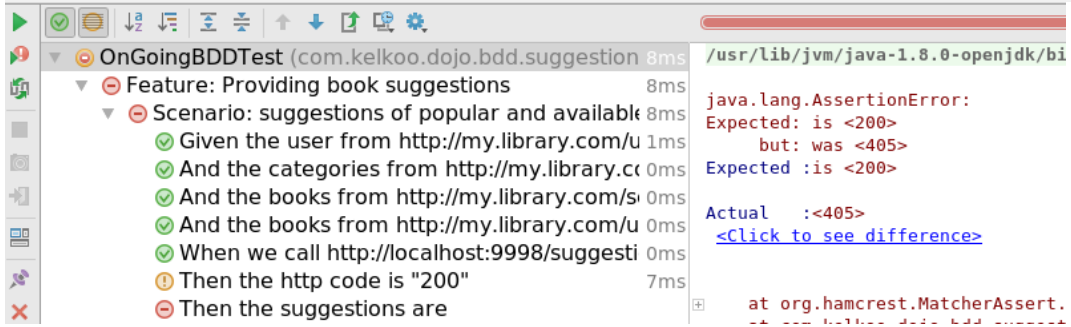
```
@When("^we call ([^\"]*)$")
public void when_we_call_suggestions_ws(String suggestionsUrl) throws Throwable {
    wsSuggestionsResponse = client.resource(suggestionsUrl).accept( ...types: "application/xml").get(ClientResponse.class);
}
```



Library | Make scenario runnable

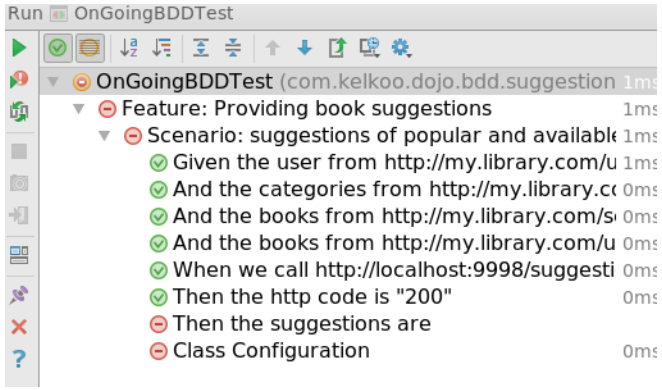



```
@Then( "^the http code is \" ([^\" ]*)\" $" )
public void the_http_code_is(Integer httpCode) throws Throwable {
    assertThat(wsSuggestionsResponse.getStatus(), is(httpCode));
}
```



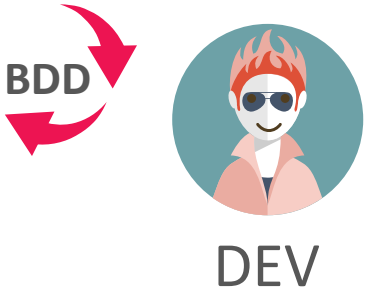
Main code does not exist…

```
@GET
@Produces("application/xml")
public Suggestions getSuggestions(@QueryParam("userId") String userId, @QueryParam("maxResults") Integer maxResults) {
        return new Suggestions();
}
```



Check results

Library | Make scenario runnable



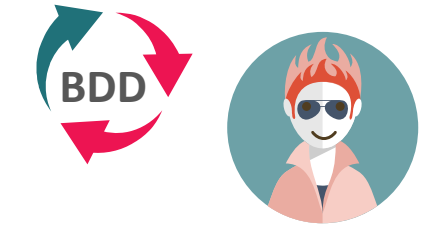

```
@Then("^the suggestions are$")
public void then_the_suggestions_are(List<Suggestion> expectedSuggestions) throws Throwable {
        SuggestionsMarshaller suggestionsMarshaller = new SuggestionsMarshaller();
        Suggestions actualSuggestions = suggestionsMarshaller.deserialize(wsSuggestionsResponse.getEntity(String.class));
        checkSameSuggestions(actualSuggestions, expectedSuggestions);
}
```

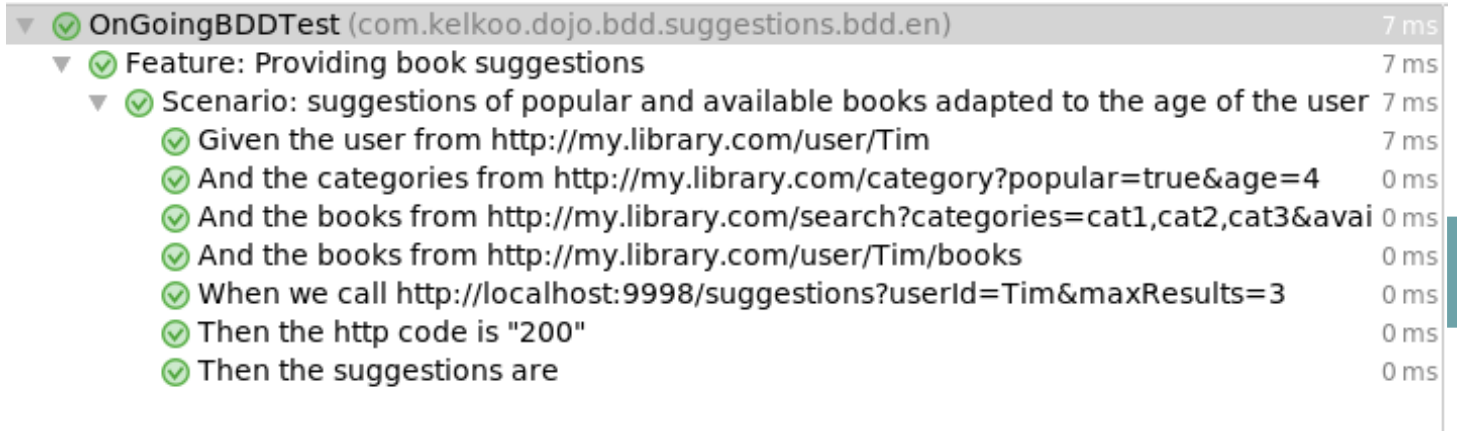
```
        java.lang.AssertionError:
        Expected: <2>
             but: was <0>
        Expected :<2>

        Actual   :<0>
```

Let's write the real code

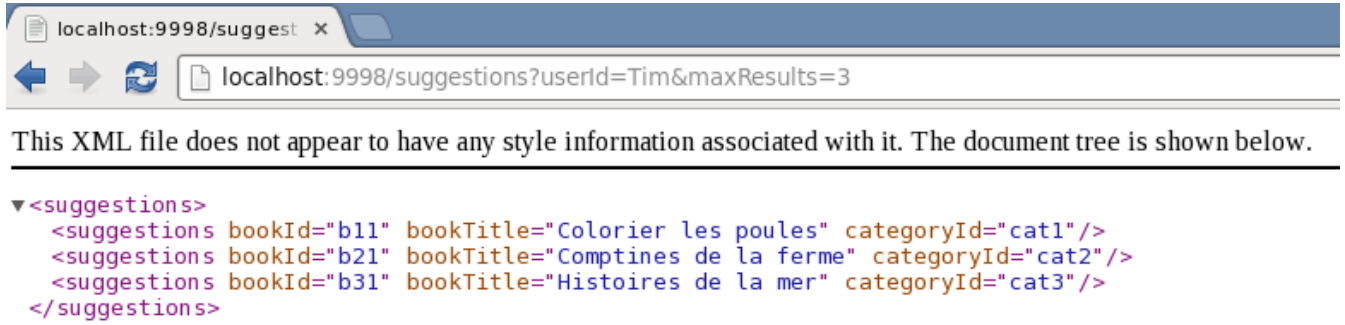
```
@GET
@Produces("application/xml")
public Suggestions getSuggestions(@QueryParam("userId") String userId) {

        Suggestions suggestions = new Suggestions();

        User user = userWSClient.retrieveUser(userId);
        Boolean isPopular = true;
        List<Category> popularCategories = categoriesWSClient.retrieveCategories(isPopular, user.getAge());
        Boolean bookAvailable = true;
        List<Book> books = searchWSClient.searchBooks(bookAvailable, extractCategoryIds(popularCategories));

        suggestions.addSuggestionsAsBooks(books);
        return suggestions;
}
```


Library | Make scenario runnable



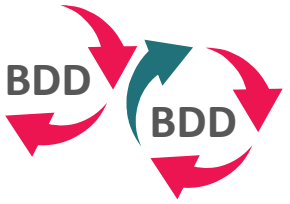


First implemented scenario!



The code is activated in the production conditions

Library | Make scenario runnable



DEV

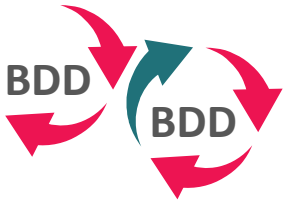
```
@level_1_specification @nominal_case @ongoing
Scenario: suggestions of popular and available books adapted to the age of the user
  Given the user "Tim"
  And he is "4" years old
  And the popular categories for this age are
    | categoryId | categoryName    |
    | cat1       | Walt Disney     |
    | cat2       | Picture books   |
    | cat3       | Bedtime stories |
  And the available books for categories "cat1,cat2,cat3" are
    | bookId | bookTitle                | categoryId |
    | b11    | Peter Pan                | cat1       |
    | b21    | Picture book about farm  | cat2       |
    | b31    | The tortoise and the hare | cat3       |
  When we ask for "3" suggestions
  Then the suggestions are
    | bookId | bookTitle                | categoryId |
    | b11    | Peter Pan                | cat1       |
    | b21    | Picture book about farm  | cat2       |
    | b31    | The tortoise and the hare | cat3       |
```

```
▼ OnGoingBDDTest (com.kelkoo.dojo.bdd.suggestions.bdd.en)
  ▼ Feature: Providing book suggestions
    ▼ Scenario: suggestions of popular and available books adapted to the age of the user
        Given the user "Tim"
        And he is "4" years old
        And the popular categories for this age are
        And the available books for categories "cat1,cat2,cat3" are
        When we ask for "3" suggestions
        Then the suggestions are

You can implement missing steps with the snippets below:

@Given("^the user \"([^\"]*)\"$")
public void the_user(String arg1) throws Throwable {
    // Express the Regexp above with the code you wish you had
    throw new PendingException();
}
```

```
@Given("^the user \"([^\"]*)\"$")
public void given_the_user(String userId) throws Throwable {
    user.setUserId(userId);
    given_the_user_from_user_ws( this.user.getUserId(), new UserStep(user).fields   );
}

@Given("^he is \"([^\"]*)\" years old$")
public void given_he_is_years_old(Integer age) throws Throwable {
    user.setAge(age);
    given_the_user_from_user_ws( user.getUserId(), new UserStep(user).fields   );
}
@Given("^the popular categories for this age are$")
public void given_the_popular_categories_for_this_age_are(List<Category> popularCategoriesGivenAgeUser)
        throws Throwable {
    Boolean isPopular = true ;
    given_the_categories_from_categories_ws(isPopular, user.getAge(), popularCategoriesGivenAgeUser);
}
```

Reusing executable steps with a lower level of abstraction

Library | Make scenario runnable



DEV

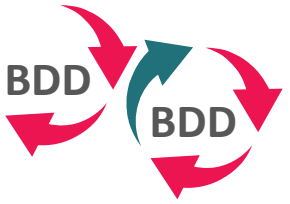
```
@level_1_specification @nominal_case @ongoing
Scenario: suggestions of popular and available books adapted to the age of the user
  Given the user "Tim"
  And he is "4" years old
  And the popular categories for this age are
    | categoryId | categoryName    |
    | cat1       | Walt Disney     |
    | cat2       | Picture books   |
    | cat3       | Bedtime stories |
  And the available books for categories "cat1,cat2,cat3" are
    | bookId | bookTitle                | categoryId |
    | b11    | Peter Pan                | cat1       |
    | b21    | Picture book about farm  | cat2       |
    | b31    | The tortoise and the hare | cat3       |
  When we ask for "3" suggestions
  Then the suggestions are
    | bookId | bookTitle                | categoryId |
    | b11    | Peter Pan                | cat1       |
    | b21    | Picture book about farm  | cat2       |
    | b31    | The tortoise and the hare | cat3       |
```

- OnGoingBDDTest (com.kelkoo.dojo.bdd.suggestions.bdd.en)
  - Feature: Providing book suggestions
    - Scenario: suggestions of popular and available books adapted to the age of the user
      - Given the user "Tim"
      - And he is "4" years old
      - And the popular categories for this age are
      - And the available books for categories "cat1,cat2,cat3" are
      - When we ask for "3" suggestions
      - Then the suggestions are
    - Scenario: limit the number of suggestions
      - Given the user "Tim"
      - And he is "4" years old
      - And the popular categories for this age are
      - And the available books for categories "cat1,cat2,cat3" are
      - When we ask for "2" suggestions
      - Then the suggestions are
      - Class Configuration

```
@GET
@Produces("application/xml")
public Suggestions getSuggestions(@QueryParam("userId") String userId, @QueryParam("maxResults") Integer maxResults) {

        Suggestions suggestions = new Suggestions();
        maxResults = maxResults == null ? DEFAULT_MAX_RESULT : maxResults;

        User user = userWSClient.retrieveUser(userId);
        Boolean isPopular = true;
        List<Category> popularCategories = categoriesWSClient.retrieveCategories(isPopular, user.getAge());
        Boolean bookAvailable = true;
        List<Book> booksForSuggestions = searchWSClient.searchBooks(bookAvailable, extractCategoryIds(popularCategories));

        // Reduce number of results
        if (booksForSuggestions.size() > maxResults) {
            booksForSuggestions = booksForSuggestions.subList( 0, maxResults);
        }

        suggestions.addSuggestionsAsBooks(booksForSuggestions);
        return suggestions;
}
```

Reusing steps

Library | Make scenario runnable



DEV

```
@level_1_specification @nominal_case @ongoing
Scenario:  limit the number of suggestions
  Given the user "Tim"
  And he is "4" years old
  And "3" books are available on popular categories for his age
  When we ask for "2" suggestions
  Then "2" suggestions are proposed from the previous books
```

- OnGoingBDDTest (com.kelkoo.dojo.bdd.suggestions.bdd.en)
  - Feature: Providing book suggestions
    - Scenario: limit the number of suggestions
      - Given the user "Tim"
      - And he is "4" years old
      - And "3" books are available on popular categories for his age
      - When we ask for "2" suggestions
      - Then "2" suggestions are proposed from the previous books

```
@Given("^\"([^\"]*)\" books are available on popular categories for his age$")
public void books_are_available_on_popular_categories_for_his_age(int nbBooks) throws Throwable {
    given_the_popular_categories_for_this_age_are(asList( new Category( categoryId: "cat1", categoryName: "category1") )) ;
    List<Book> books = new ArrayList<~>();
    for (int i = 0; i < nbBooks; i++) {
        books.add( new Book( bookId: "b1"+i, bookTitle: "book1"+i, categoryId: "cat1" ) ) ;
    }
    given_the_search_results_for_categories_are( categoryIds: "cat1", books) ;
}

@Then("^\"([^\"]*)\" suggestions are proposed from the previous books$")
public void suggestions_are_proposed_from_the_previous_books(Integer nbSuggestions) throws Throwable {
    Suggestions suggestions = Suggestions.suggestionsFromBooks(  searchResult.subList( i: 0, nbSuggestions) ) ;
    then_the_suggestions_are(suggestions.getSuggestions()) ;
}
```

Generate data to make a scenario easier to read

- OnGoingBDDTest (com.kelkoo.dojo.bdd.suggestions.bdd.en)
  - Feature: Providing book suggestions
    - Scenario: limit the number of suggestions
      - Given the user "Tim"
      - And he is "4" years old
      - And "3" books are available on popular categories for his age
      - When we ask for "2" suggestions
      - Then "2" suggestions are proposed from the previous books

Library | Make scenario runnable

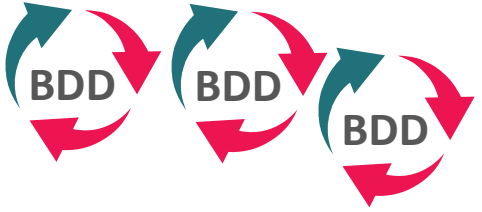
DEV

```
@level_0_high_level @nominal_case @ongoing
Scenario: providing book suggestions
  Given a user
  When we ask for suggestions
  Then the suggestions are popular and available books adapted to the age of the user
```

Implement a high level scenario

```
@Given("^a user$")
public void given_a_user() throws Throwable {
    given_the_user( userId: "userId1");
    given_he_is_years_old( age: 4);
    given_the_popular_categories_for_this_age_are(asList( new Category( categoryId: "cat1", categoryName: "category1"), new Category( categoryId: "cat2", categoryName: "category2")  ));
    given_the_search_results_for_categories_are( categoryIds: "cat1,cat2",
                                                 asList( new Book( bookId: "b11", bookTitle: "book11", categoryId: "cat1" ),
                                                         new Book( bookId: "b21", bookTitle: "book21", categoryId: "cat2" ),
                                                         new Book( bookId: "b31", bookTitle: "book31", categoryId: "cat3" )));
}

@When("^we ask for suggestions$")
public void when_we_ask_for_suggestions() throws Throwable {
    when_we_ask_for_suggestions( maxResults: 3);
}

@Then("^the suggestions are popular and available books adapted to the age of the user$")
public void then_the_suggestions_are_popular_and_available_books_adpated_to_the_age_of_the_user() throws Throwable {
    then_the_suggestions_are(asList( new Suggestion( bookId: "b11", bookTitle: "book11", categoryId: "cat1" ),
                                     new Suggestion( bookId: "b21", bookTitle: "book21", categoryId: "cat2" ),
                                     new Suggestion( bookId: "b31", bookTitle: "book31", categoryId: "cat3" )));
}
```


Library | Make scenario runnable



DEV

- TestValidBDDEn (com.kelkoo.dojo.bdd.suggestions.bdd.en)
  - Feature: Providing book suggestions
    - Scenario: suggestions of popular and available books adapted to the age of the user, he have never booked the suggestions
    - Scenario: suggestions of popular and available books adapted to the age of the user
    - Scenario: limit the number of suggestions
    - Scenario: limit the number of suggestions
    - Scenario: the user have never booked the suggestions
    - Scenario: the books are coming from different categories
    - Scenario: not enough suggestions, the books can come from the same categories
    - Scenario: unknown user, no suggestion
    - Scenario: one service on which the suggestion system is down
    - Scenario: unknown user, no suggestion
    - Scenario: one service on which the suggestion system depends on is down
    - Scenario: providing book suggestions

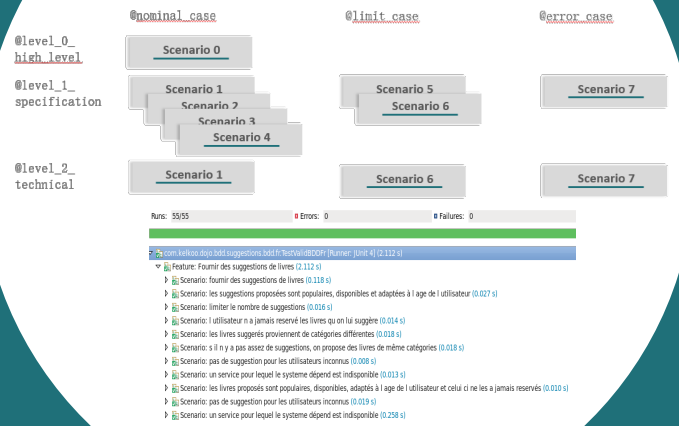
All scenarios are implemented

Library | Regression tests

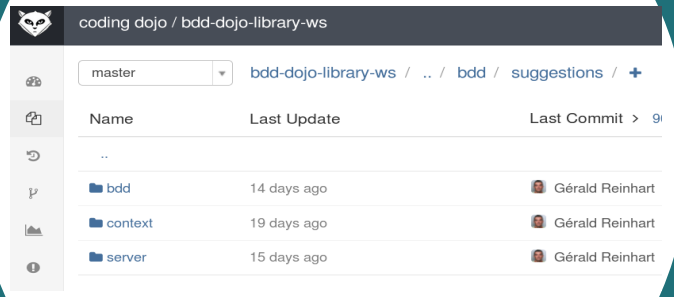
DEV

Regular regression tests

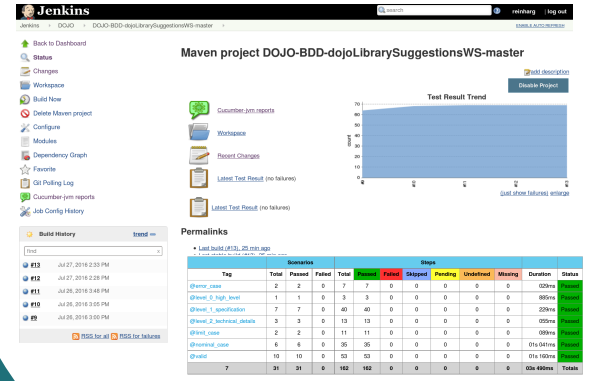
Scenarios BDD



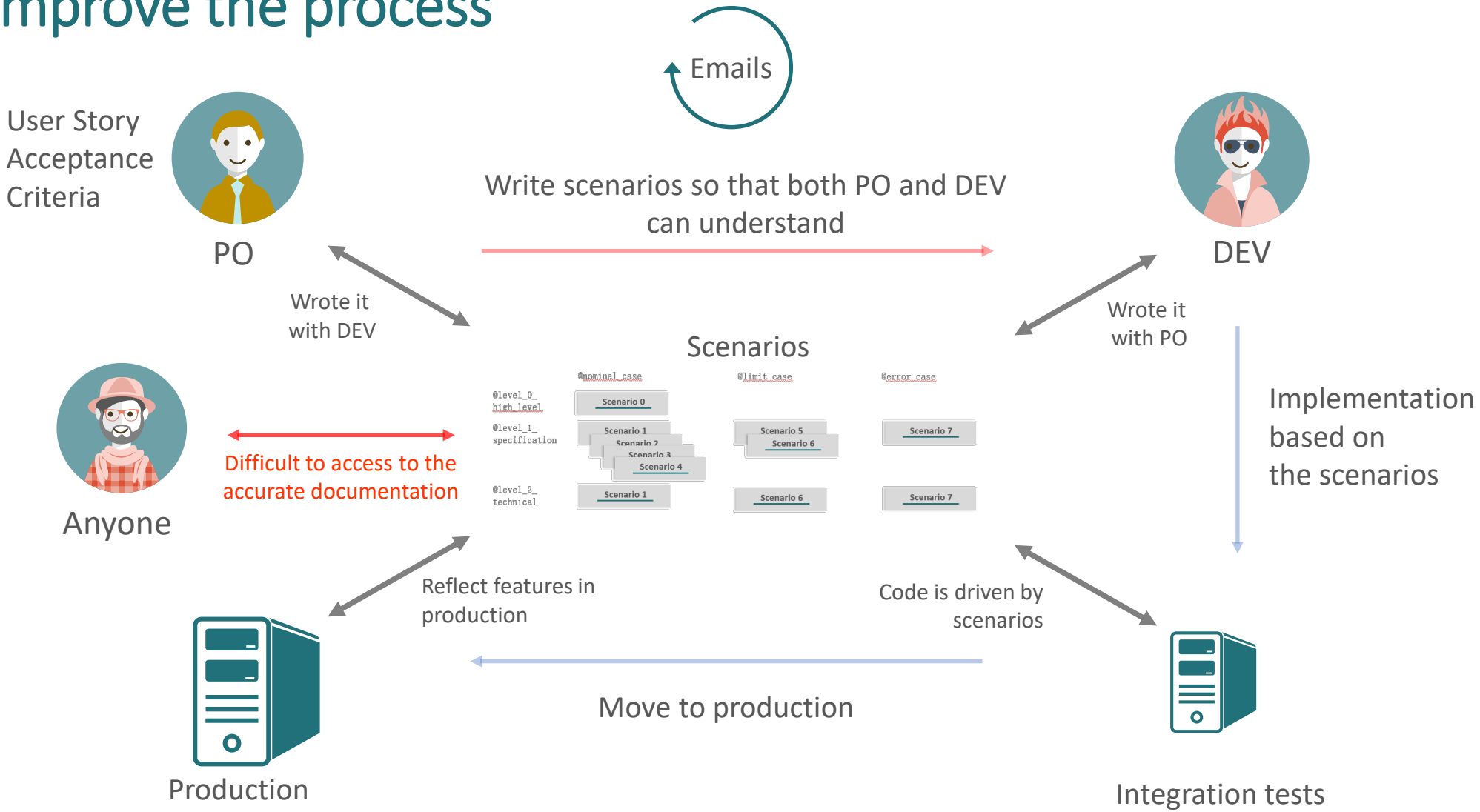
Source Code



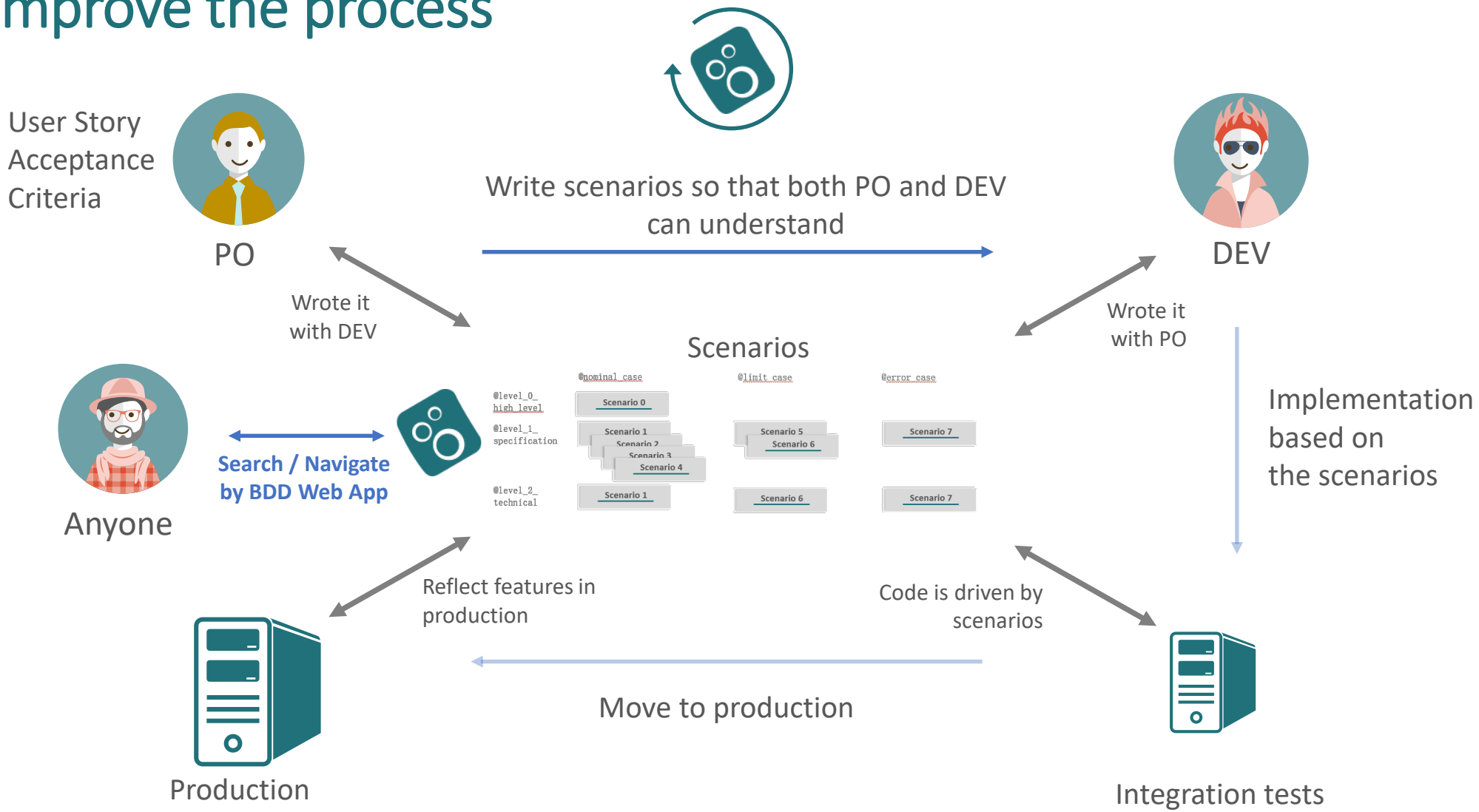
Continuous Integration



Improve the process



Improve the process



Conclusion

- Specification by example
  - Close collaboration DEV / PO is required
  - Use examples to open discussion and find many cases
  - Allows a very fast feedback loop
- Functional tests
  - Fast and stable tests
  - The developer is guided, the code is pulled by the tests
  - Flexible code is required to mock external interactions
- Runnable Documentation
  - Pulled from code, the documentation is always up to date
  - The documentation is exhaustive

Conclusion

- Specification by example
  - Close collaboration DEV / PO is required
  - Use examples to open discussion and find many cases
  - Allows a very fast feedback loop
- Functional tests
  - Fast and stable tests
  - The developer is guided, the code is pulled by the tests
  - Flexible code is required to mock external interactions
- Runnable Documentation
  - Pulled from code, the documentation is always up to date
  - The documentation is exhaustive

Conclusion

- Failure factor
  - DEV or PO not involved
  - BDD applied during the project : need to be done a the very first step
- theGardener aim is to address too important use cases not really addressed yet :
  - easily access to the scenario
  - collaborative tools to help PO and DEV discussion

    => https://github.com/KelkooGroup/theGardener/wiki

Thank you