



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

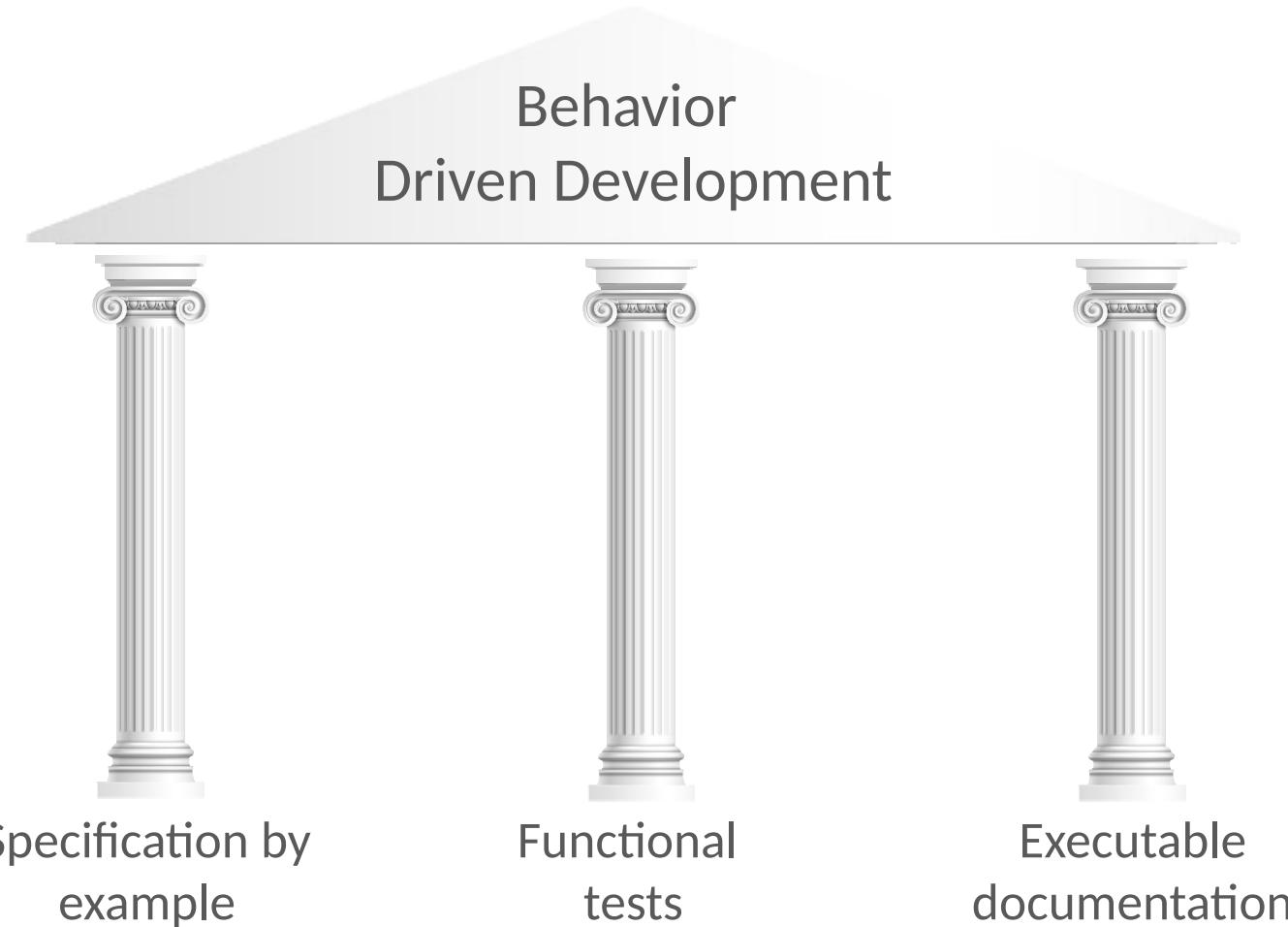


Introduction

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

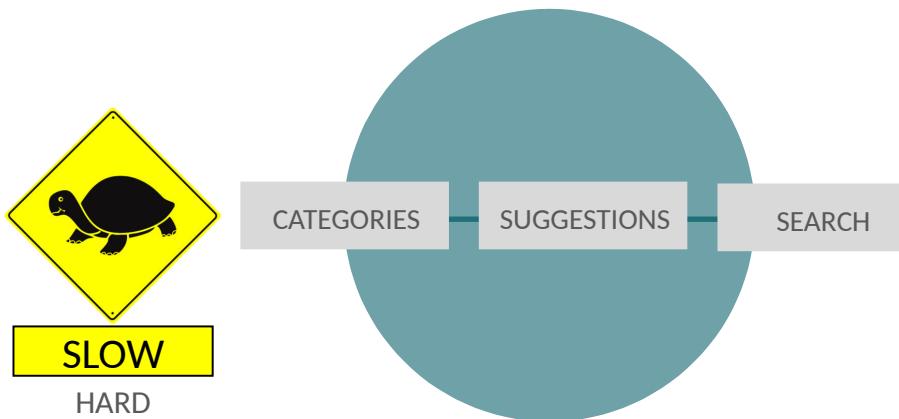
When we ask for "2" suggestions

Then the suggestions are

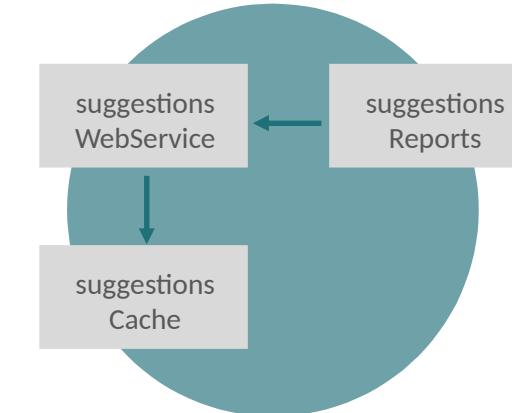
bookId	bookTitle	categoryId
lv11	Peter Pan	cat1
lv21	The tortoise and the hare	cat2



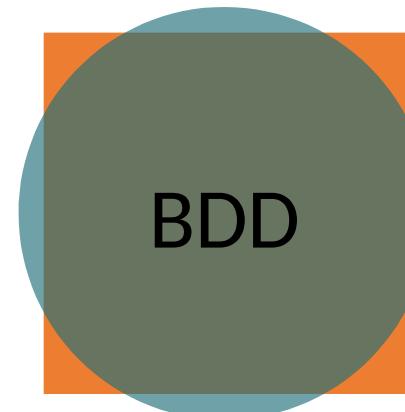
Introduction | Functional tests



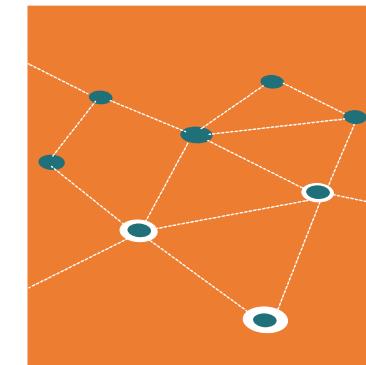
Between systems



On a system between components



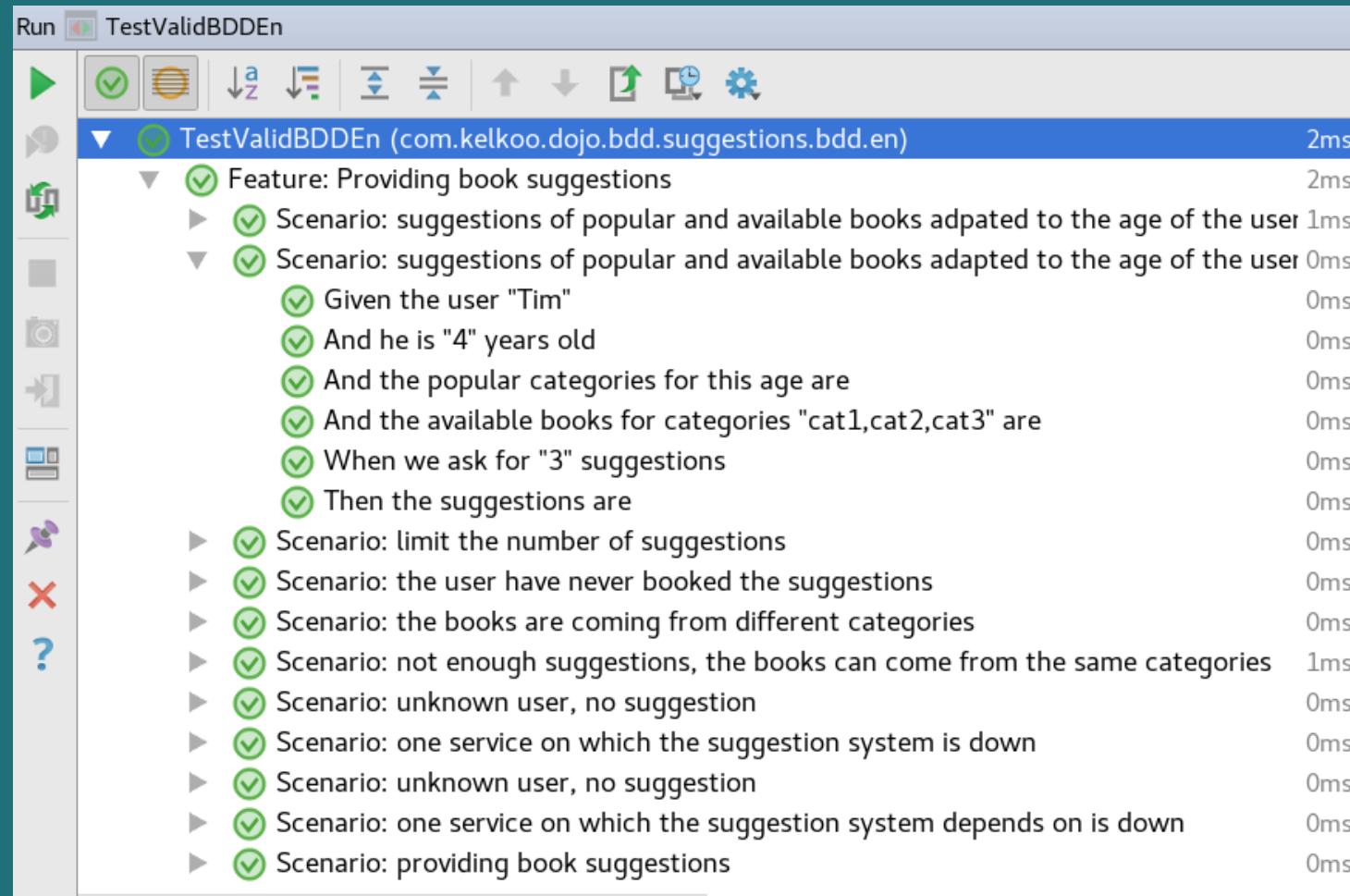
On a component



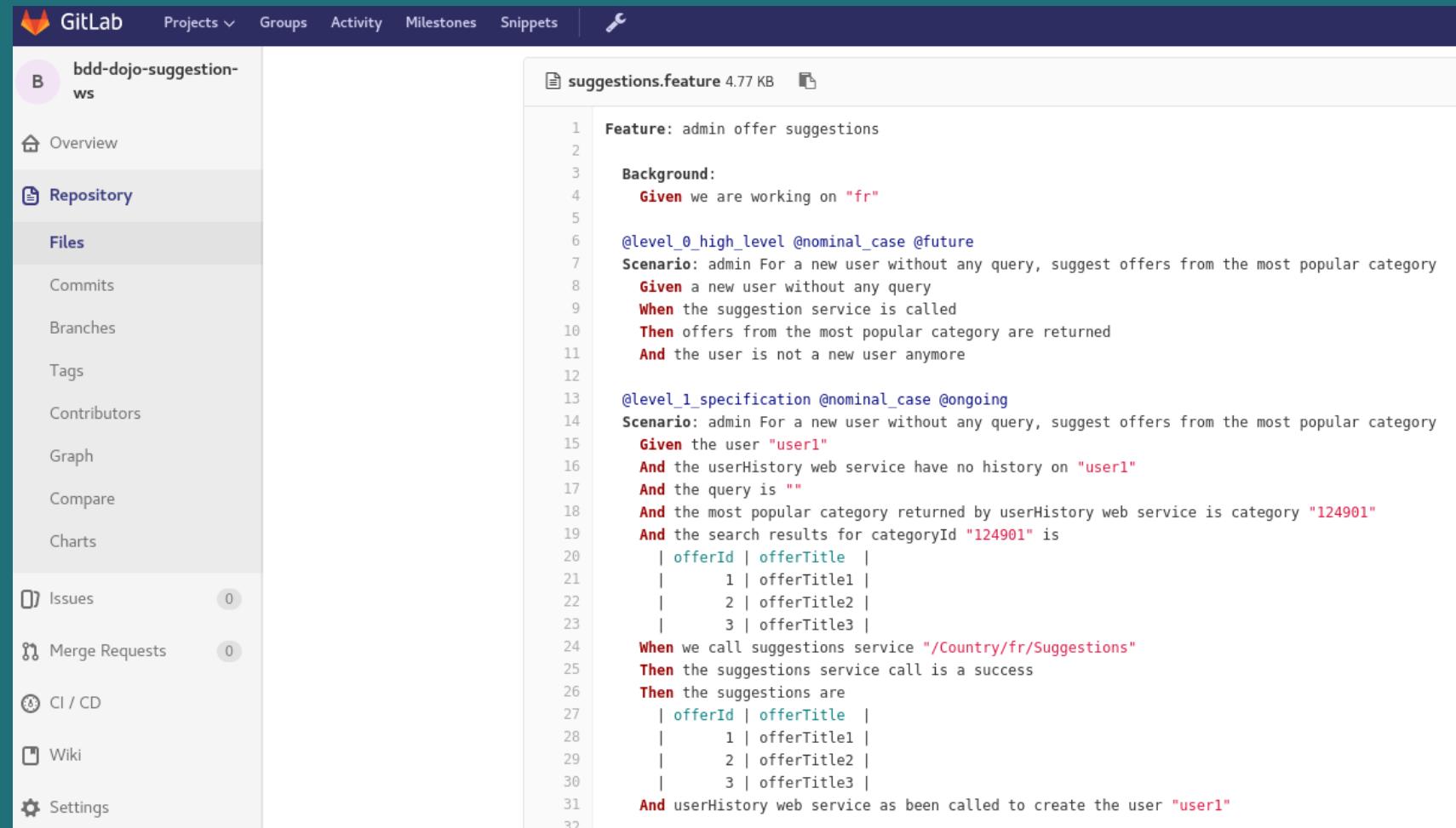
On a class



Introduction | Functional tests



Introduction | Executable Documentation



The screenshot shows a GitLab project interface for a repository named "bdd-dojo-suggestion-ws". The left sidebar lists various project sections: Overview, Repository (selected), Files, Commits, Branches, Tags, Contributors, Graph, Compare, Charts, Issues (0), Merge Requests (0), CI / CD, Wiki, and Settings. The main content area displays a file named "suggestions.feature" with a size of 4.77 KB. The file contains BDD (Behavior-Driven Development) steps in Gherkin syntax:

```
Feature: admin offer suggestions
Background:
  Given we are working on "fr"
@level_0_high_level @nominal_case @future
Scenario: admin For a new user without any query, suggest offers from the most popular category
  Given a new user without any query
  When the suggestion service is called
  Then offers from the most popular category are returned
  And the user is not a new user anymore

@level_1_specification @nominal_case @ongoing
Scenario: admin For a new user without any query, suggest offers from the most popular category
  Given the user "user1"
  And the userHistory web service have no history on "user1"
  And the query is ""
  And the most popular category returned by userHistory web service is category "124901"
  And the search results for categoryId "124901" is
    | offerId | offerTitle |
    | 1       | offerTitle1 |
    | 2       | offerTitle2 |
    | 3       | offerTitle3 |
  When we call suggestions service "/Country/fr/Suggestions"
  Then the suggestions service call is a success
  Then the suggestions are
    | offerId | offerTitle |
    | 1       | offerTitle1 |
    | 2       | offerTitle2 |
    | 3       | offerTitle3 |
  And userHistory web service as been called to create the user "user1"
```



Library | User Story to be implemented



PO



DEV



Library | User Story to be implemented

CATEGORIES

Categories of books,
popular categories by age

SUGGESTIONS

Provides book
suggestions



User

USERS

Users, ages,
books already red...

BOOKING

Booking service,
Available books

SEARCH

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



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: 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

Too technical, I don't understand

Non-speaking example

Not linked to the user

Missing 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

Missing limit number of suggestions

And the available books for categories "cat1,cat2,cat3" are

bookId	title
lv11	Peter Pan
lv21	Picture book about farm
lv31	The tortoise and the hare

Missing : never read

When we ask for "3" suggestions

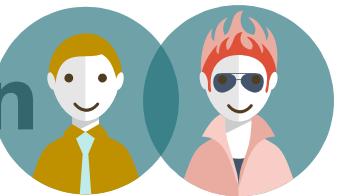
Then the suggestions are

bookId	title
lv11	Peter Pan
lv21	Picture book about farm
lv31	The tortoise and the hare

Missing : 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

Missing : never read

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 "2" suggestions

Then the suggestions are

bookId	title	categoryId
lv11	Peter Pan	cat1
lv21	Picture book about farm	cat2

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

Missing : never read

And the available books for categories "cat1,cat2,cat3" are

bookId	title	categoryId
lv11	Peter Pan	cat1
lv12	Pinocchio	cat1
4	lv21	cat2
lv31	The tortoise and the hare	cat3

When we ask for "2" suggestions

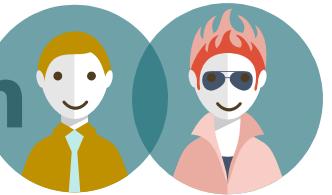
Then the suggestions are

bookId	title	categoryId
lv11	Peter Pan	cat1
lv21	Picture book about farm	cat2

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

What are we testing ?
Prefer several scenario

And the available books for categories "cat1,cat2,cat3" are

bookId	title	categoryId
lv11	Peter Pan	cat1
lv12	Pinocchio	cat1
lv13	Bamby	cat1
lv21	Picture book about farm	cat2
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 "2" suggestions

Then the suggestions are

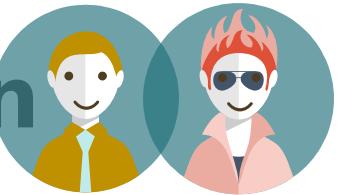
bookId	title	categoryId
lv21	Picture book about farm	cat2
lv31	The tortoise and the hare	cat3

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

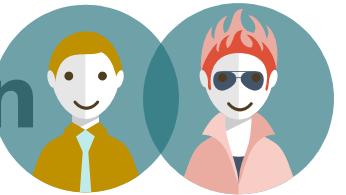
bookId	title	categoryId
lv11	Peter Pan	cat1
lv21	Picture book about farm	cat2

*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

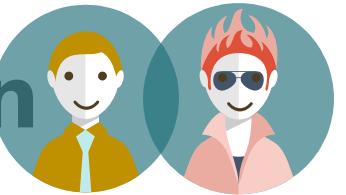
bookId	title	categoryId
lv11	Peter Pan	cat1

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



PO DEV

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 user "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 technical 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 0

Scenario 5

Scenario 7

Scenario 3

Scenario 4

Scenario 6

Scenario 7

Scenario 1

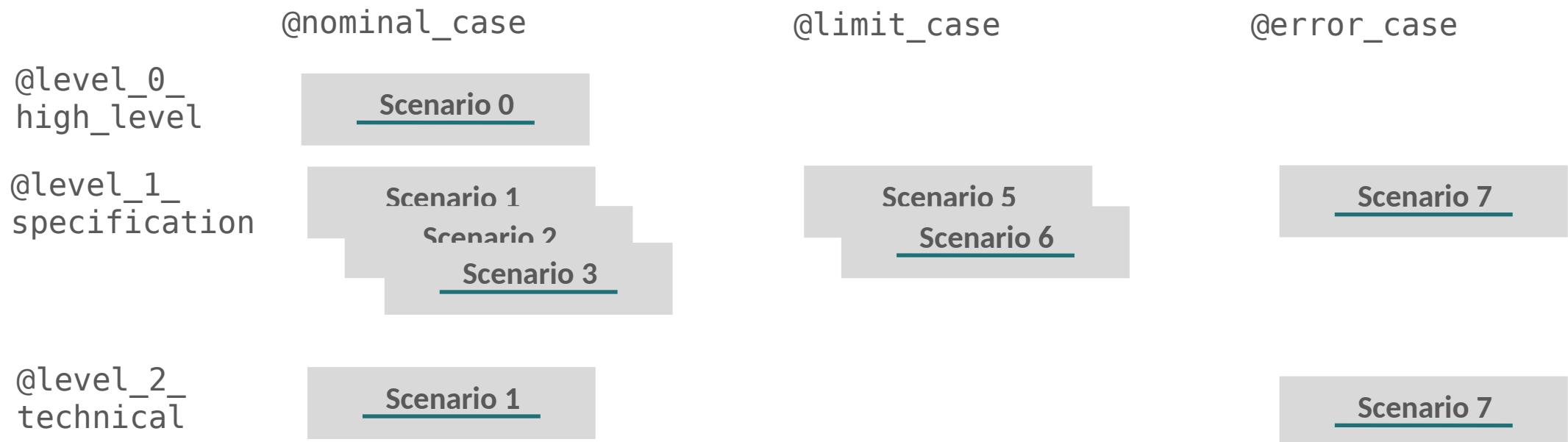


Library | Organize scenarios

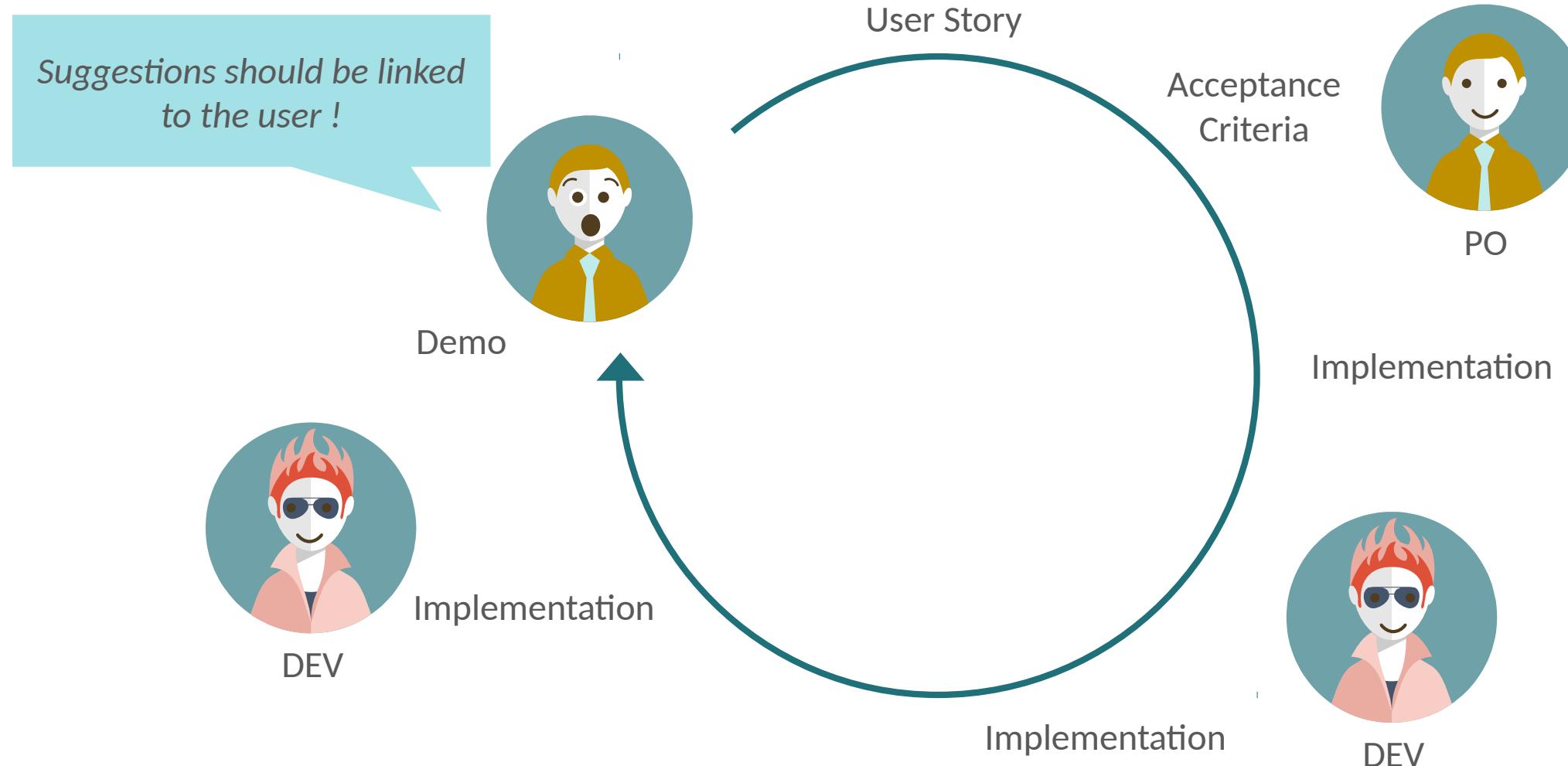


PO DEV

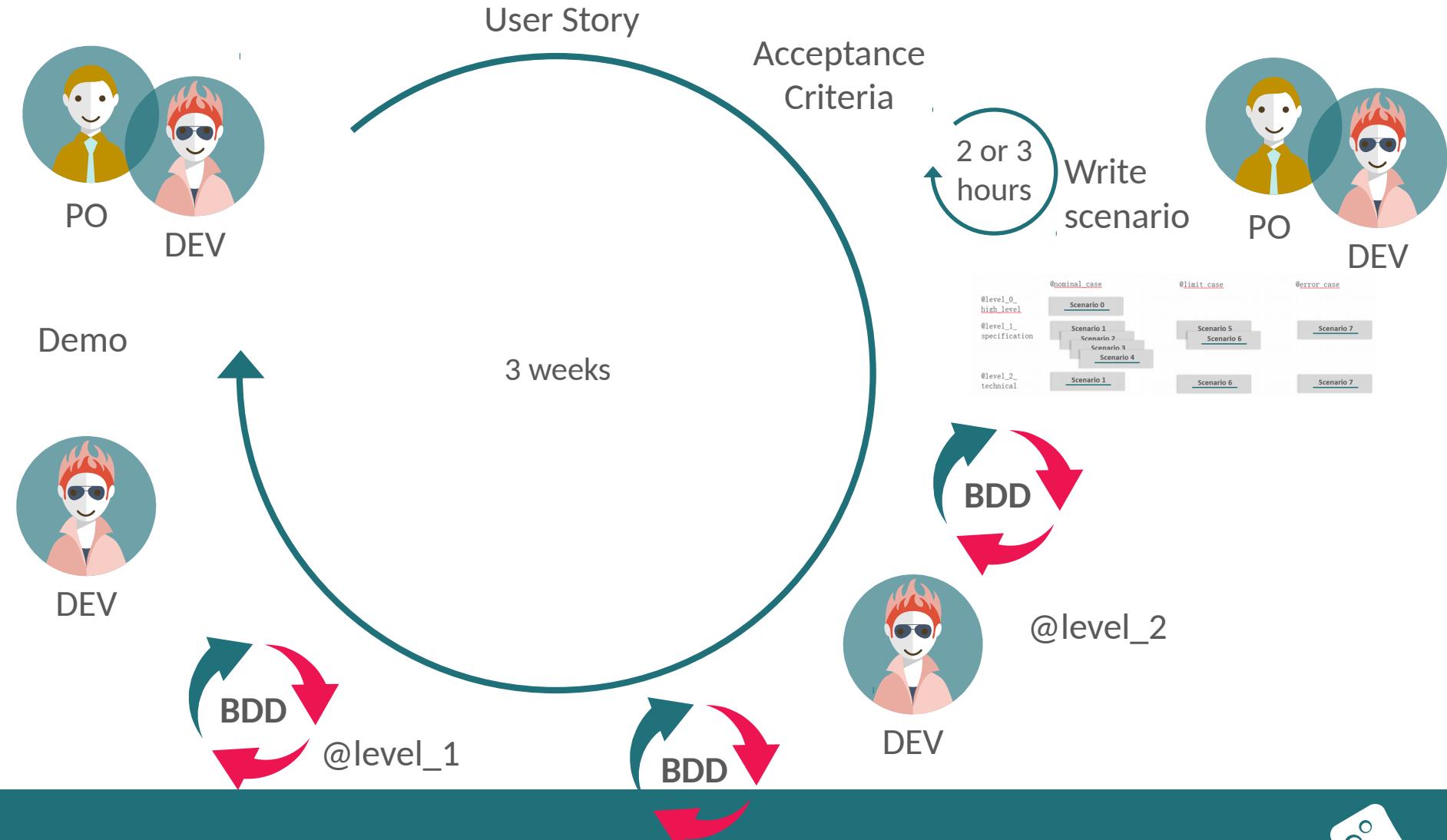
As a user of the library,
I **wish to** book suggestions
to make discoveries



Library | Without specification by example



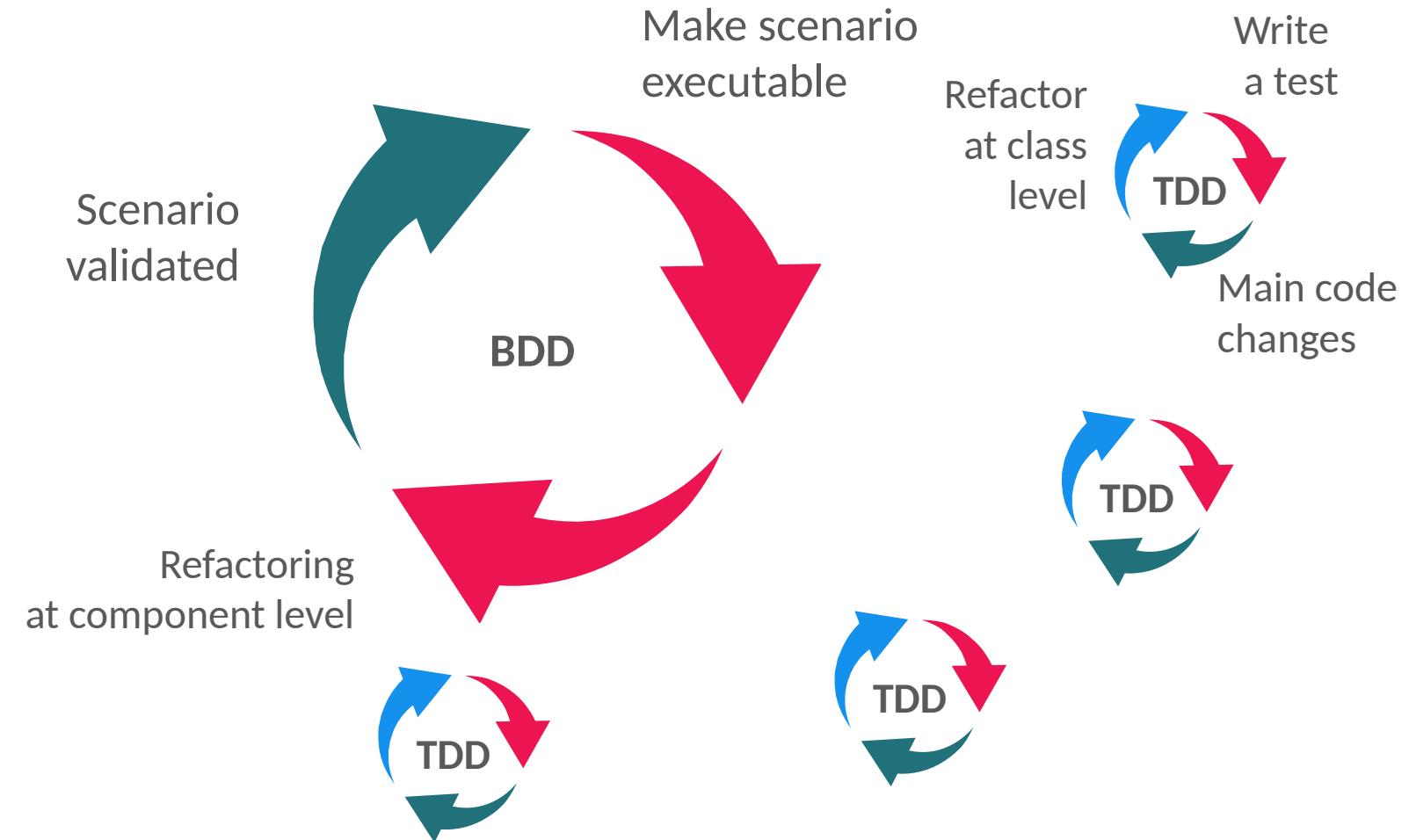
Library | Sprint roadmap



Library | Development cycles



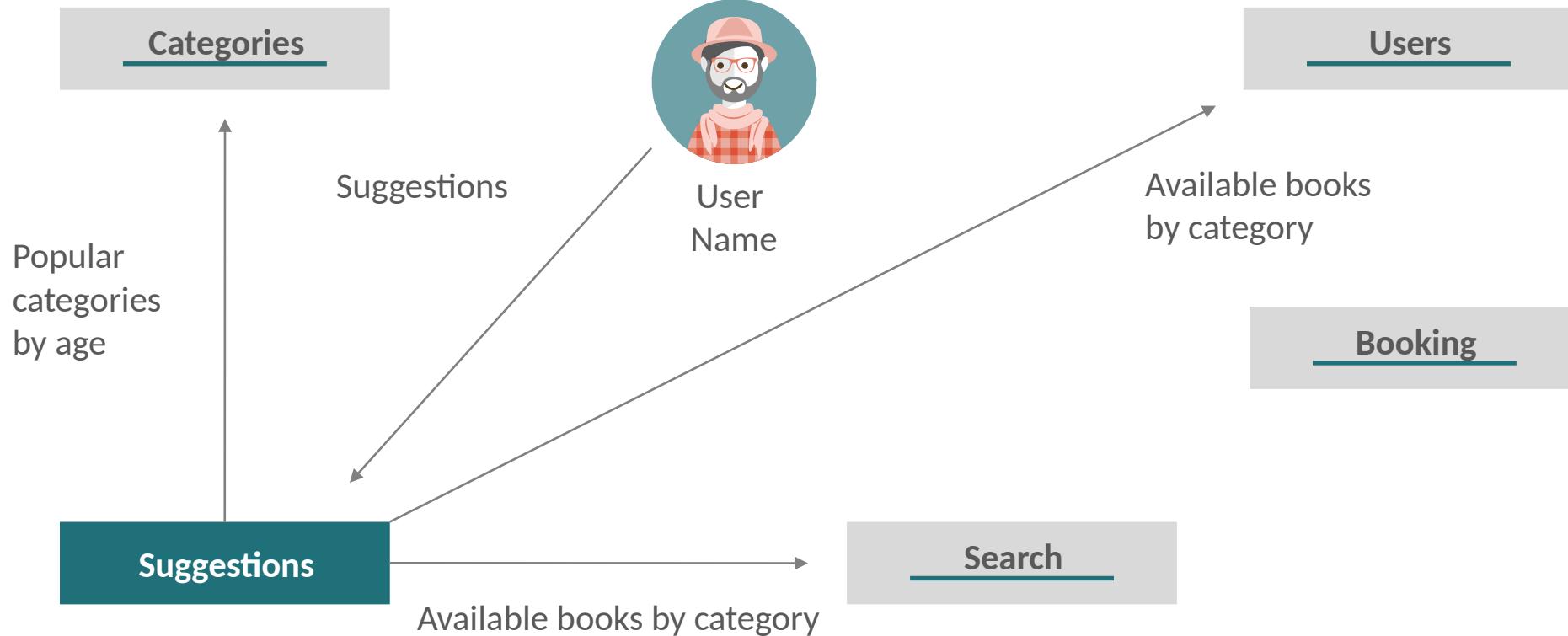
DEV



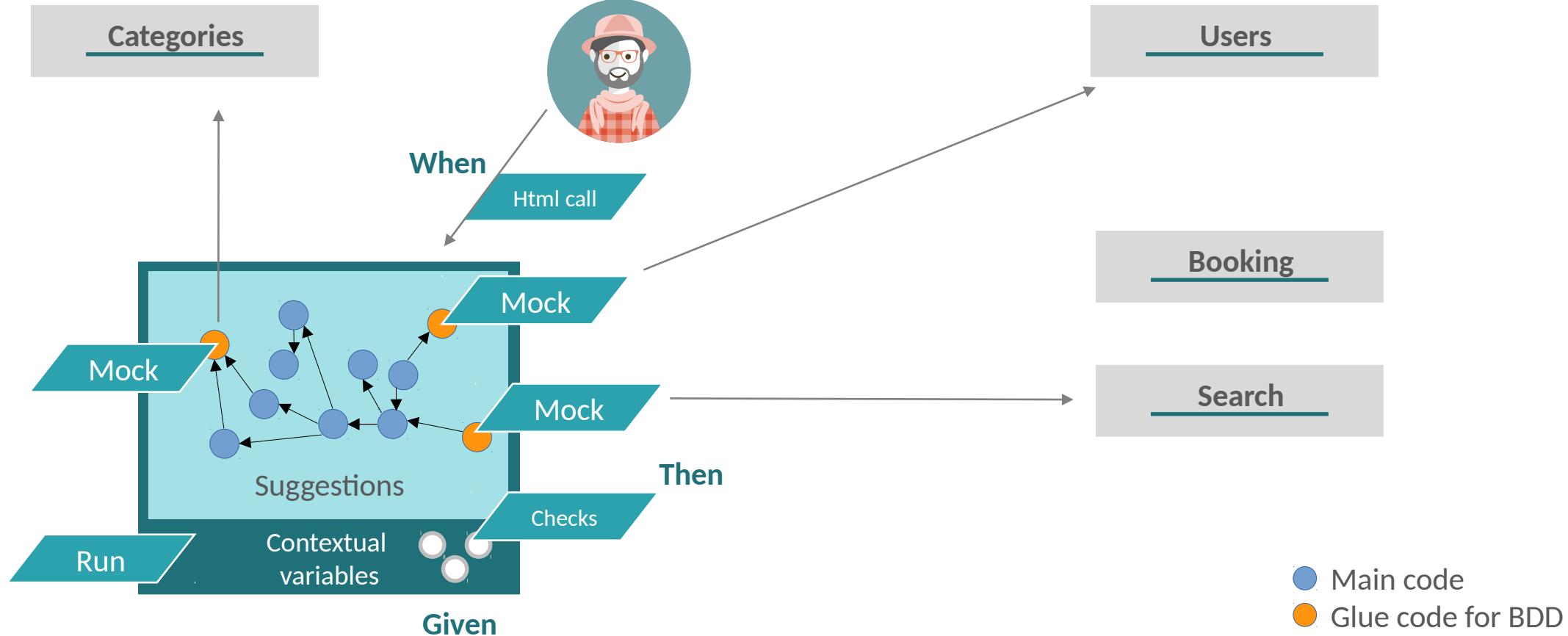
Library | Make scenario executable



DEV



Library | Make scenario runnable



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

```
@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);
}
```

*Glue code between steps
and main code*

```
▼ user = {User@3760} "User [userId=Tim, age=4, alreadyBookedBooks=[]]"
  ▶ f userId = "Tim"
  ▶ f age = {Integer@3820} 4
  f alreadyBookedBooks = {ArrayList@3821} size = 0
```

Contextual variable



Library | Make scenario runnable

BDD



DEV

The screenshot shows the IntelliJ IDEA interface with the 'Debugger' tab selected in the top navigation bar. Below it, the 'Console' tab is also visible. The main pane displays the 'OnGoingBDDTest' class from 'com.kelkoo dojo.bdd.suggestions'. The class contains a single method: 'given_the_categories_from_categories_ws'. This method has annotations: '@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()))'. The 'when' block uses 'categoriesWSClientMock' as a mock object.

```
@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=false
| 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()));
}

Run OnGoingBDDTest

```

The screenshot shows the 'Run' tool window in IntelliJ IDEA. It lists the 'OnGoingBDDTest' run configuration. The 'OnGoingBDDTest' class is shown with its test methods: 'given_the_categories_from_categories_ws' and 'Scenario: suggestions of popular and available books adapted to the age'. The 'given_the_categories_from_categories_ws' method is expanded, showing its annotations and code. The 'when' block uses 'categoriesWSClientMock' as a mock object.

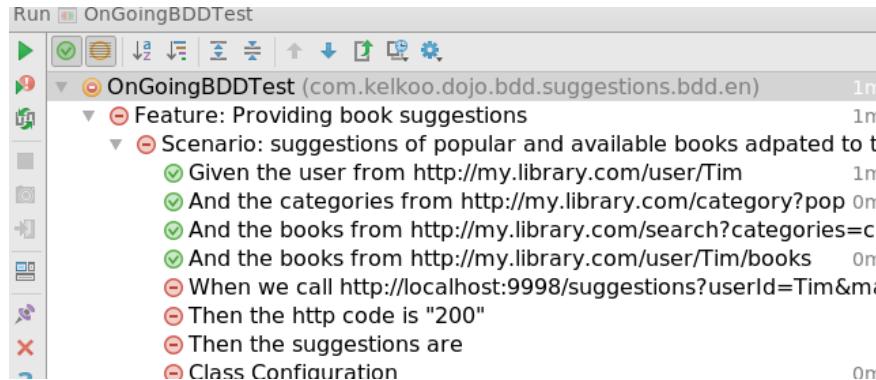


Library | Make scenario runnable

BDD

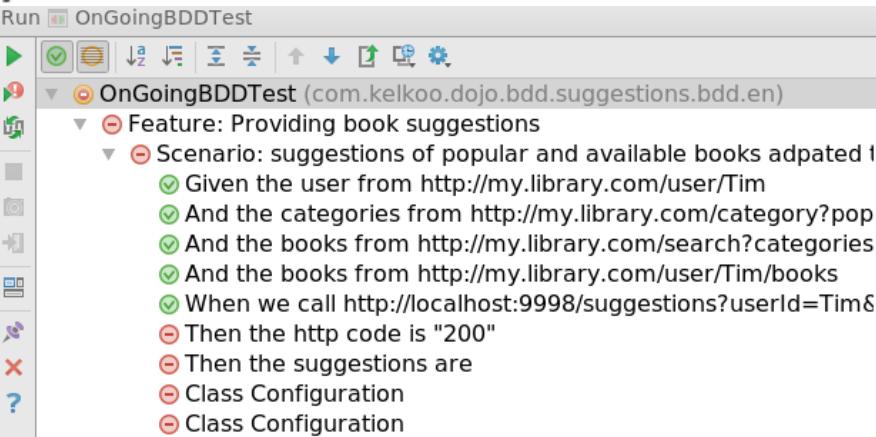


DEV



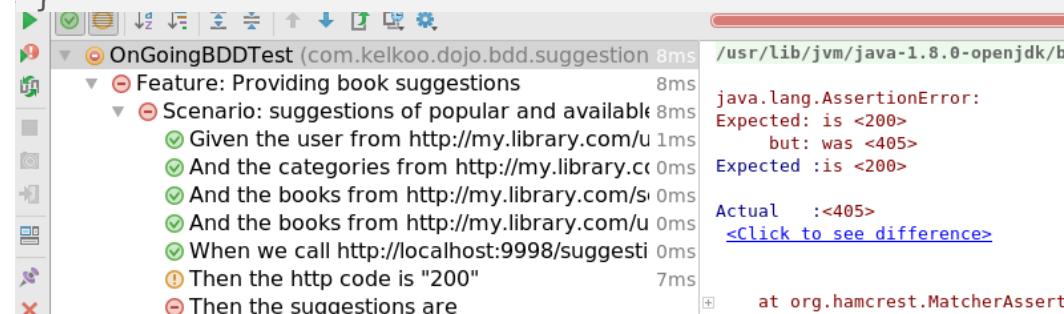
Main code call

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

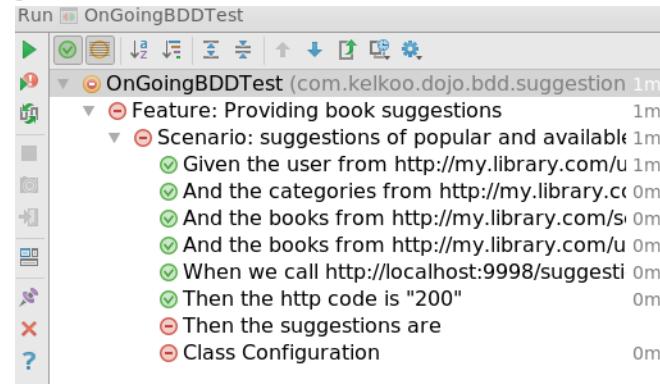


Library | Make scenario runnable

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



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



Main code does not exist...

Check results



DEV



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



DEV

▼	✔ OnGoingBDDTest (com.kelkoo dojo.bdd.suggestions.bdd.en)	7 ms
▼	✔ Feature: Providing book suggestions	7 ms
▼	✔ Scenario: suggestions of popular and available books adapted to the age of the user	7 ms
✔ Given the user from http://my.library.com/user/Tim	7 ms	
✔ And the categories from http://my.library.com/category?popular=true&age=4	0 ms	
✔ And the books from http://my.library.com/search?categories=cat1,cat2,cat3&avai	0 ms	
✔ And the books from http://my.library.com/user/Tim/books	0 ms	
✔ When we call http://localhost:9998/suggestions?userId=Tim&maxResults=3	0 ms	
✔ Then the http code is "200"	0 ms	
✔ Then the suggestions are	0 ms	

First implemented scenario!

localhost:9998/suggest x

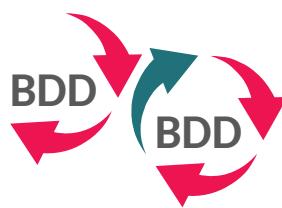
localhost:9998/suggestions?userId=Tim&maxResults=3

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<suggestions>
  <suggestions bookId="b11" bookTitle="Colorier les poules" categoryId="cat1"/>
  <suggestions bookId="b21" bookTitle="Comptines de la ferme" categoryId="cat2"/>
  <suggestions bookId="b31" bookTitle="Histoires de la mer" categoryId="cat3"/>
</suggestions>
```

The code is activated in the production conditions





Library | Make scenario runnable

```
@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)          On
  ▼ ⓘ Feature: Providing book suggestions                            On
    ▼ ⓘ Scenario: suggestions of popular and available books adapted to the age of the user On
      ⓘ Given the user "Tim"                                         On
        ⓘ And he is "4" years old                                     On
        ⓘ And the popular categories for this age are                  On
        ⓘ And the available books for categories "cat1,cat2,cat3" are On
        ⓘ When we ask for "3" suggestions                           On
        ⓘ Then the suggestions are                                On
      You can implement missing steps with the snippets below:
      @Given("^the user \"([^\"]*)\"$")
      public void given_the_user(String arg1) throws Throwable {
        // Express the Regexp above with the code you wish you had
        throw new PendingException();
      }
```

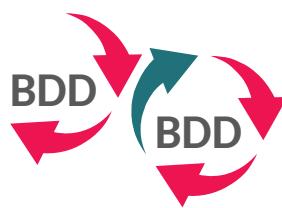
```
@Given("^the user \"([^\"]*)\"$")
public void given_the_user(String userId) throws Throwable {
  user.setUserId(userId);
  given_the_user_from_user_ws( this.user.getUserId(), new UserStep(user).fields );
}

@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

```
@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      |

@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;
}
```

▼ ⓘ 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

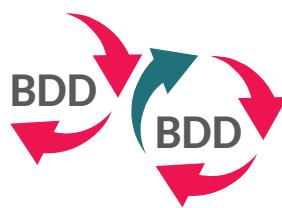
✓ When we ask for "2" suggestions

○ Then the suggestions are

ⓘ Class Configuration

Reusing steps





Library | Make scenario runnable

```
@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
```

```
@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 );
}

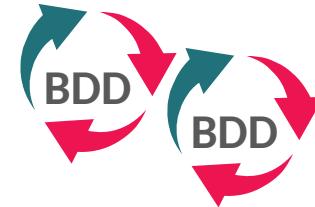
@Then("^\"([^\"]*)\" suggestions are proposed from the previous books$")
public void suggestions_are_proposed_from_the_previous_books(Integer nbSuggestions) throws Throwable {
    Suggestions suggestions = Suggestions.suggestionsFromBooks( searchResult.subList( 0, nbSuggestions ) );
    then_theSuggestionsAre(suggestions.getSuggestions());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

```
▼ ⓘ 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
```

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
```





DEV

Library | Make scenario runnable

```
@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" )) );
}

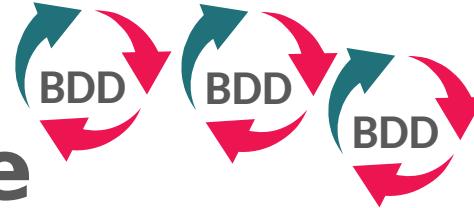
@When("^we ask for suggestions$")
public void when_we_ask_for_suggestions() throws Throwable {
    when_we_ask_for_suggestions( maxResults: 3);
}

@Then("^the suggestions are popular and available books adapted to the age of the user$")
public void then_theSuggestions_are_popular_and_available_books_adpated_to_the_age_of_the_user() throws Throwable {
    then_theSuggestions_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

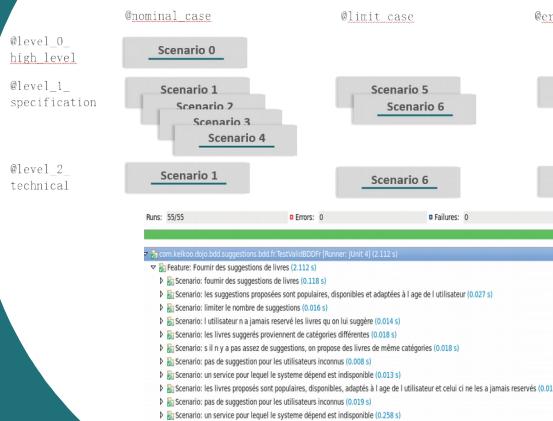


Library | Regression tests



DEV

Scenarios BDD



Regular regression tests

Source Code



Continuous Integration

Jenkins

Maven project DOJO-BDD-dojoLibrarySuggestionsWS-master

Test Result Trend

Build	Start Time	Duration	Status
#13	Jul 27, 2014 2:23 PM	0:09m	Success
#12	Jul 27, 2014 2:28 PM	0:09m	Success
#11	Jul 26, 2014 3:58 PM	0:09m	Success
#10	Jul 26, 2014 3:50 PM	0:09m	Success
#9	Jul 26, 2014 3:00 PM	0:09m	Success
#8	Jul 26, 2014 2:59 PM	0:09m	Success
#7	Jul 26, 2014 2:58 PM	0:09m	Success
#6	Jul 26, 2014 2:57 PM	0:09m	Success
#5	Jul 26, 2014 2:56 PM	0:09m	Success
#4	Jul 26, 2014 2:55 PM	0:09m	Success
#3	Jul 26, 2014 2:54 PM	0:09m	Success
#2	Jul 26, 2014 2:53 PM	0:09m	Success
#1	Jul 26, 2014 2:52 PM	0:09m	Success
#0	Jul 26, 2014 2:51 PM	0:09m	Success

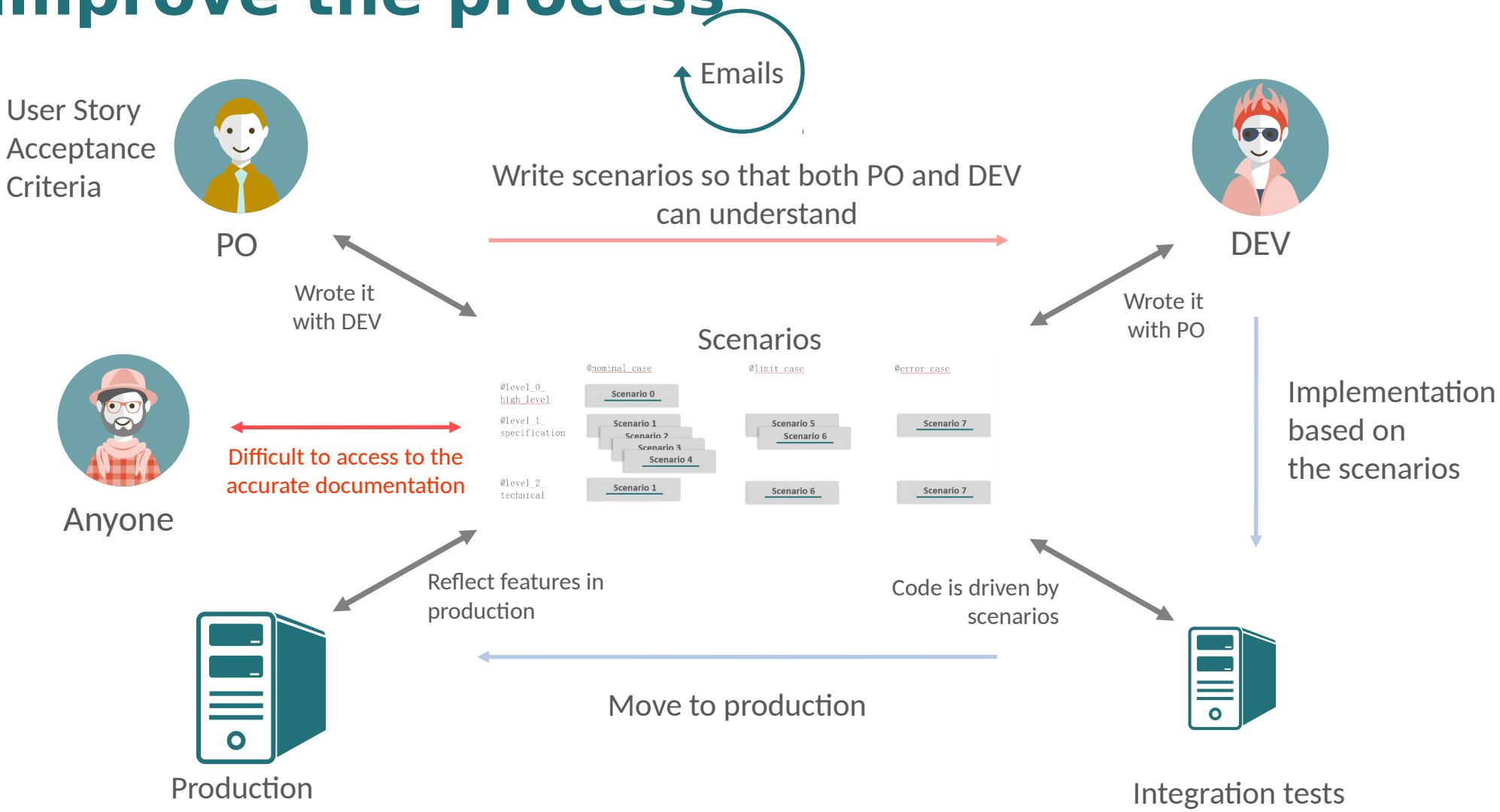
Permalinks

- Last build (#13), 25 min ago
- Build History
- Scenarios
- Steps

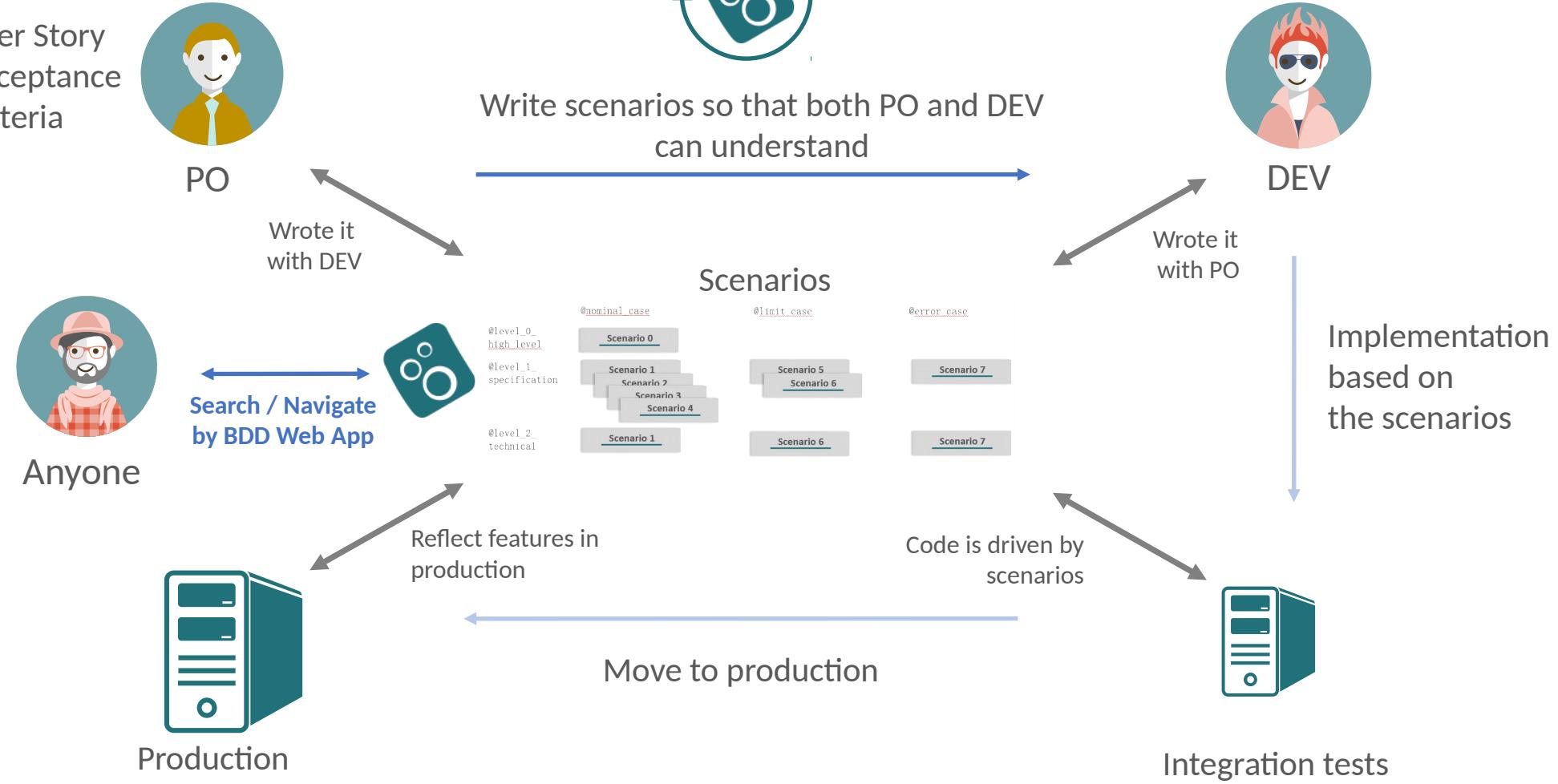
Tag	Total	Passed	Failed	Skipped	Pending	Undefined	Missing	Duration	Status
@error_case	2	2	0	7	0	0	0	0:09m	Success
@level_0_high_level	1	1	0	3	0	0	0	0:09m	Success
@level_1_low_level	7	7	0	49	0	0	0	0:09m	Success
@bdd	2	2	0	13	0	0	0	0:09m	Success
@bdd	2	2	0	11	0	0	0	0:09m	Success
@bdd	6	6	0	35	0	0	0	0:09m	Success
@bdd	10	10	0	53	0	0	0	0:09m	Success
Total	31	31	0	162	0	0	0	0:26h	Success



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 at 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

