

UCAAT 2014 - Munich

Arnaud Bouzy
Product Owner

arnaud.bouzy@smartesting.com

Acceptance testing in DevOps projects



Our long and
painful journey
towards DevOps

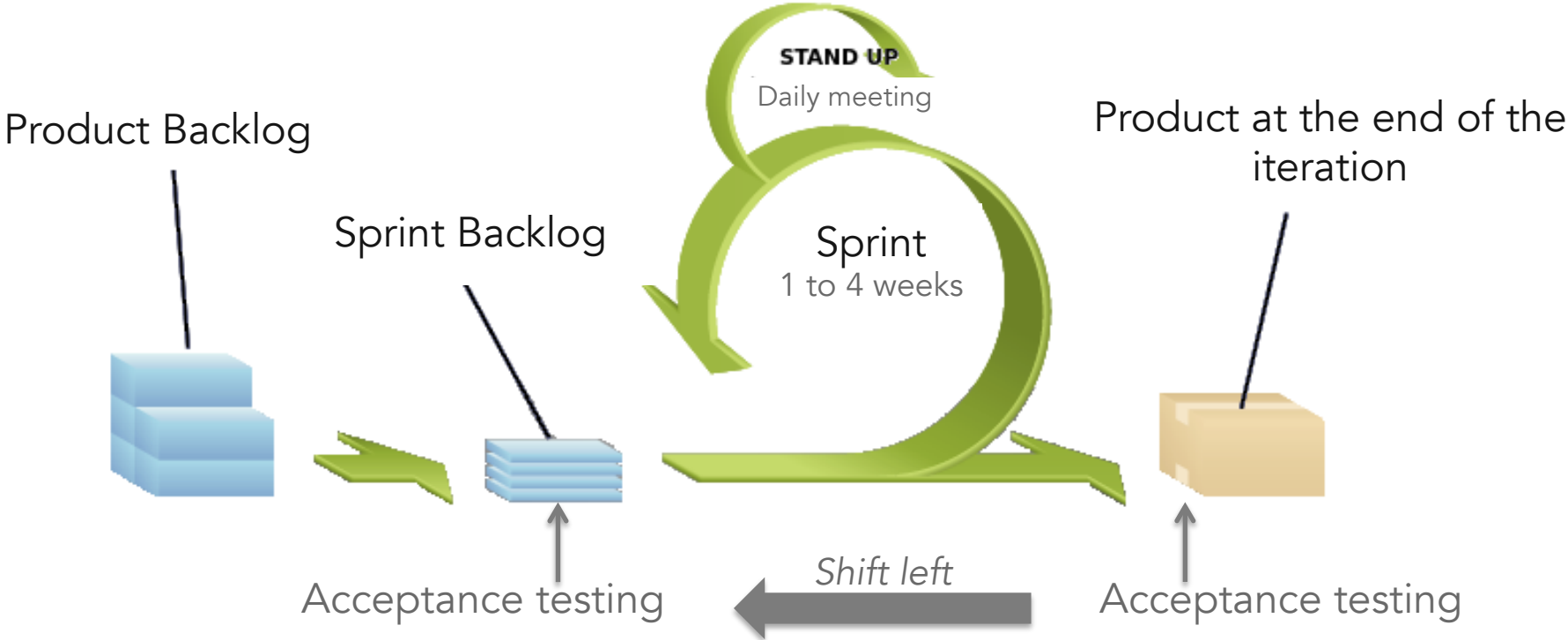
What we've learnt from testing standpoint

- Shorter iterations (1 to 4 weeks) lead to massive test automation completed by exploratory testing
- Acceptance tests become the specification
- Testing starts earlier in the development process: *Shift left!*



Acceptance testing
driven development

Shift left

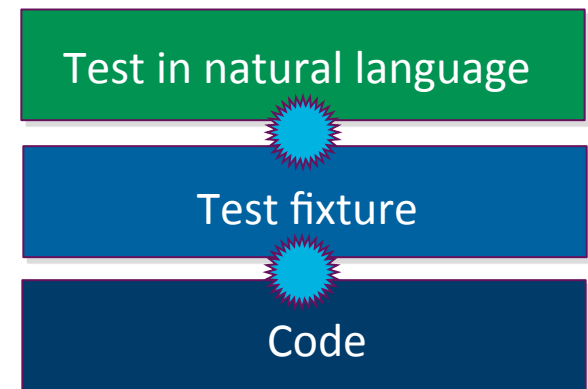


Scrum team



Acceptance Testing Driven Development (ATDD)

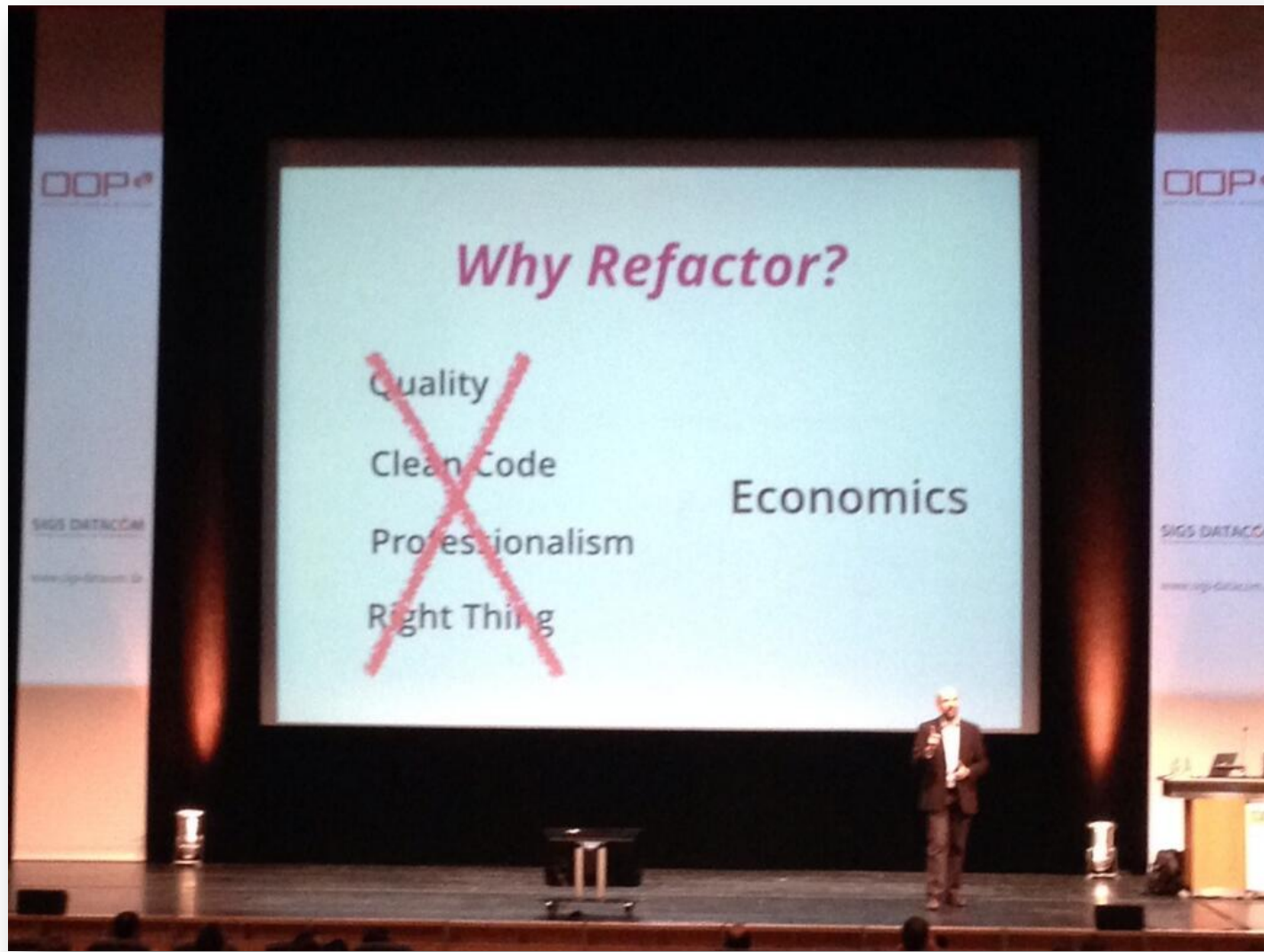
- Acceptance test is a powerful artifact to improve communication
- Test as the definition of 'STOP'
- Written prior to development by tester
- Based on a business DSL (domain specific Language)
- Confirmed with stakeholders
- Mostly automated



Acceptance Testing Driven Development (ATDD)

- Benefits:
 - Improve communication and collaboration: crucial with distributed team
 - Shared understanding of what a successful implementation means
 - Better coverage of business expectations
 - Faster feed back
- Challenges:
 - New methodology that requires rigor and discipline
 - Find the right balance between people/process/tool

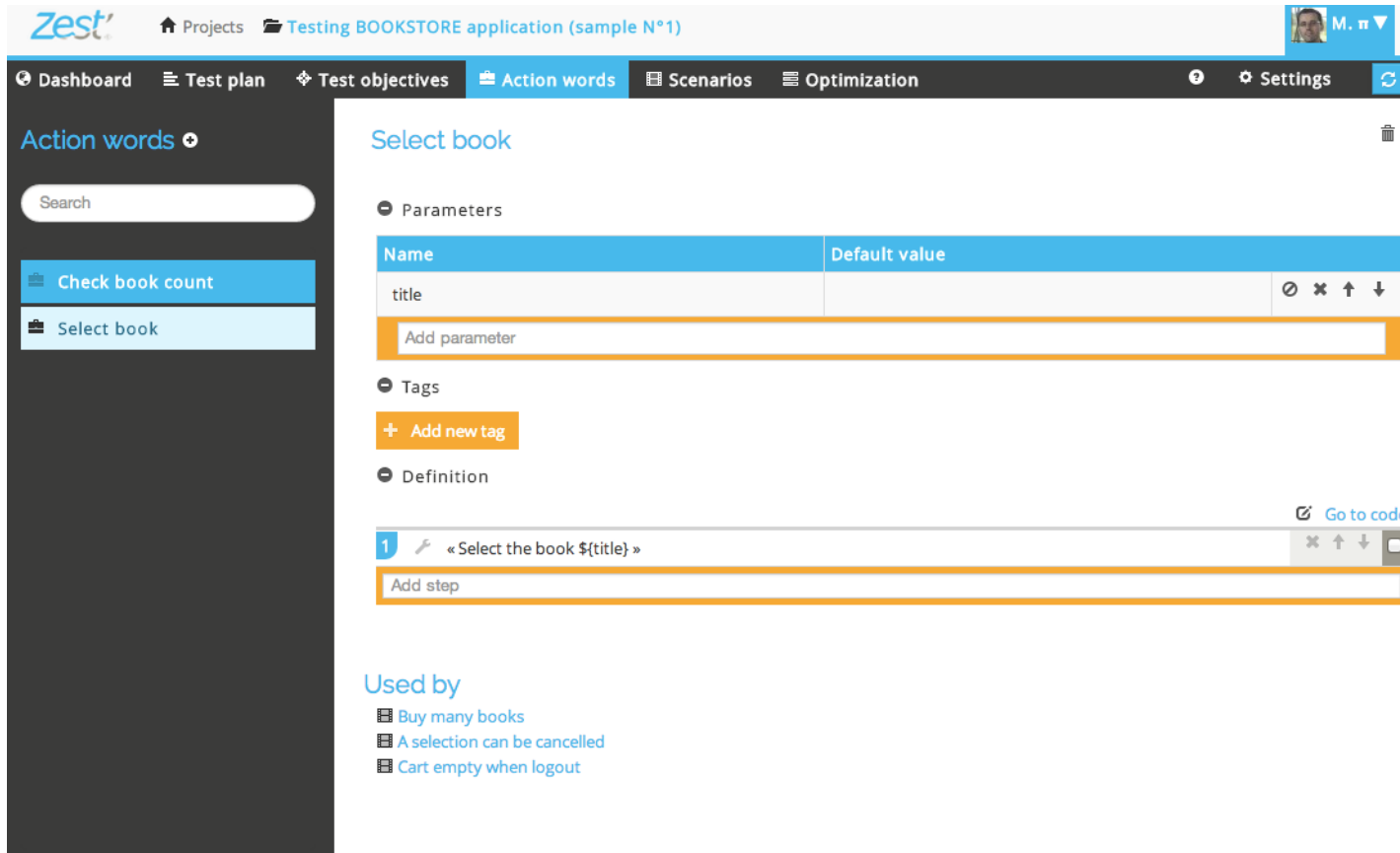
ATDD & Refactoring



Martin Fowler

Using a DSL to
define acceptance
tests and doing
refactoring

Define new business actions...



The screenshot displays the Zest application interface. The top navigation bar includes 'Dashboard', 'Test plan', 'Test objectives', 'Action words', 'Scenarios', 'Optimization', 'Settings', and a user profile icon. The left sidebar shows 'Action words' with a search bar and two buttons: 'Check book count' and 'Select book'. The main content area is titled 'Select book' and contains the following sections:

- Parameters:** A table with columns 'Name' and 'Default value'. The first row contains 'title'. Below the table is an 'Add parameter' input field.
- Tags:** An 'Add new tag' button.
- Definition:** A code editor showing the definition: « Select the book \${title} ». Below it is an 'Add step' input field.
- Used by:** A list of scenarios: 'Buy many books', 'A selection can be cancelled', and 'Cart empty when logout'.



Define progressively your business terminology with Action Words. Enable collaboration based on acceptance tests.

...or define business actions right from the tests

The screenshot displays the Zest testing tool interface. The top navigation bar includes 'Dashboard', 'Test plan', 'Test objectives', 'Action words', 'Scenarios', and 'Optimization'. The left sidebar shows a search bar and a list of scenarios: 'A selection can be cancelled', 'Buy many books', and 'Cart empty when logout'. The main area shows the 'Buy many books' scenario with a description: 'This test is an end-to-end test from book selection to payment'. A modal dialog titled 'Promote steps as action word' is open, allowing the user to define an action word for a specific step. The dialog includes a text input for the action word name, a 'Parameters' section with a table, and a 'Sources' section with a list of test steps.

| Parameters | Steps | Sources |
|------------|------------------------------------|--|
| title | 1 « Log with default account » | <input checked="" type="checkbox"/> « Cart empty when logout » |
| | 2 « Check that user is logged » | <input checked="" type="checkbox"/> « Buy many books » |
| | 3 « Go to on-line library » | <input checked="" type="checkbox"/> « A selection can be cancelled » |
| | 4 « Select book » title = title | |
| | 5 « Add to cart » | |

 Define progressively your business terminology with Action Words. Enable collaboration based on acceptance tests.

Reuse, reuse and reuse!

The screenshot displays the Zest! web application interface. The top navigation bar includes 'Dashboard', 'Test plan', 'Test objectives', 'Action words', 'Scenarios', and 'Optimization'. The 'Scenarios' tab is active, showing a scenario definition with seven steps:

- 1 « Log with default account »
- 2 « Check that user is logged »
- 3 « Go to on-line library »
- 4 Select book
 - title = « Harry Potter and the order of the phoenix »
- 5 « Add to cart »
- 6 Select book
 - title = « Harry Potter and the goblet of fire »
- 7 « Add to cart »

A dropdown menu is open below step 4, showing suggestions for the selected book. The suggestions are:

- Create action word
- Create result
- Create action
- Select book
- Check that user is logged
- Check that user is logged
- Check that user is logged
- Check that user is logged

The word 'Suggestions' is overlaid in orange text on the dropdown menu. The interface also shows a search bar and a list of scenarios on the left sidebar.



Create and maintain consistent scenarios for your project

Add, remove, modify Action Words and Scenarios

The screenshot displays the Zest application interface. The top navigation bar includes 'Dashboard', 'Test plan', 'Test objectives', 'Action words', 'Scenarios', and 'Optimization'. The left sidebar shows 'Action words' with a search bar and two buttons: 'Check book count' and 'Select book'. The main content area is titled 'Select book' and contains several sections:

- Parameters:** A table with columns 'Name' and 'Default value'. The first row contains 'title'. Below the table is an 'Add parameter' input field. An arrow points from the text 'Add parameters to Action Word' to the 'title' parameter.
- Tags:** An 'Add new tag' button.
- Definition:** A code editor showing the definition: « Select the book \${title} ». Below it is an 'Add step' input field. An arrow points from the text 'Propagate automatically to the scenarios' to the definition.
- Used by:** A list of scenarios: 'Buy many books', 'A selection can be cancelled', and 'Cart empty when logout'. An arrow points from the text 'Propagate automatically to the scenarios' to this list.

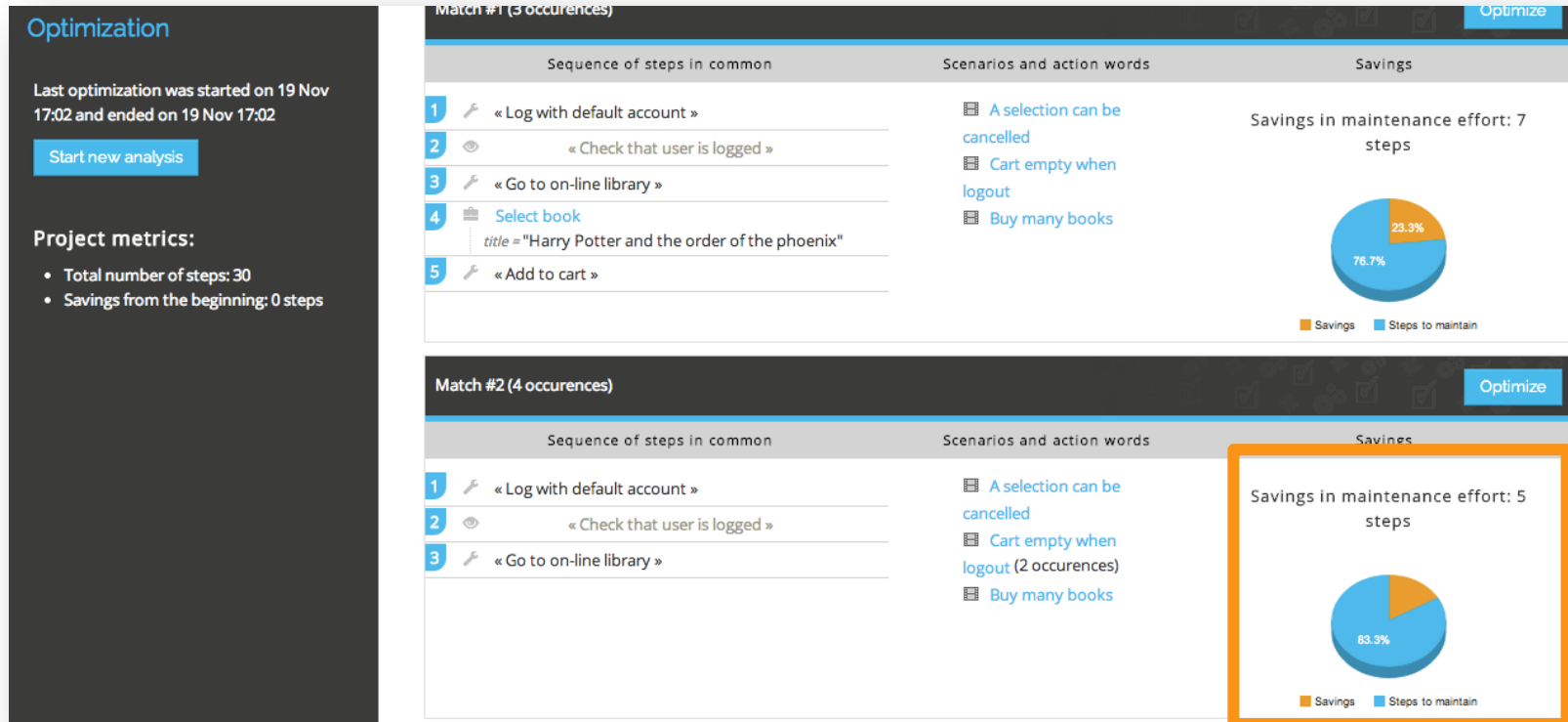
Additional annotations include a 'Go to code' link in the definition section and a trash icon in the top right corner of the main content area.

 Test refactoring enables to perform automatically impact analysis and test maintenance tasks

Evils of duplication



Analyse and optimize continuously your tests



Reduce dramatically the maintenance cost!

Generate scripts

The screenshot displays the Zest web interface for a project named "Testing BOOKSTORE application (sample N°1)". The interface is divided into several sections:

- Dashboard:** Shows project metrics including 2 action words and 3 scenarios, with a link to "Start guided tour".
- Testing BOOKSTORE application (sample N°1):** A central panel with a welcome message and a "Publication" section listing options like "XML export of project", "XML export of tests", "Excel export of tests", and "Ruby export of tests".
- Timeline:** A section showing a list of action words with their assignees and timestamps. One action word is visible: "assigne" by "v.pretre@sm" on "09 Dec 15:11".
- Code Editor:** A window titled "project_spec.rb" showing Ruby code for defining action words. The code includes comments and method calls for actions like "Schedule a meeting", "Create User-Account", and "User Edition".

```
# encoding: UTF-8
require_relative 'actionwords'

describe 'Testing D00DLE (sample N°2)' do
  before(:each) do
    @actionwords = TestingDoodleSampleN2::Actionwords.new
  end

  it 'Schedule a meeting' do
    # Meeting schedule
    # tags: Priority:1
    @actionwords.goto_doodle_website()
    @actionwords.create_an_event(title = "My meeting", location = "Office", name = "A good
meeting for everyone", email = "foo@company.com")
    @actionwords.select_date()
    @actionwords.select_time()
    @actionwords.select_poll_type(type = "Basic")
    @actionwords.send_invite()
    # TODO: implement result: "Check you have received the email from Doodle"
  end

  it 'Create User-Account' do
    # Create a new user
    # tags: Creation, Priority:1
    @actionwords.goto_doodle_website()
    @actionwords.create_account()
    @actionwords.enter_personal_information(email = "johan@example.com", name = "johan",
pass = "tiger")
    # TODO: implement result: "check you have received a confirmation by email with an
activation code"
    @actionwords.activate_the_account()
  end

  it 'User Edition' do
    # Edit a User
    # tags: Edition, Priority:2
    @actionwords.goto_doodle_website()
    @actionwords.sign_in(name = "Johan", pass = "tiger")
    @actionwords.select_manage_useraccount_menu()
    @actionwords.change_email_address(email = "johan@other.com")
    @actionwords.save()
  end
end
```

 Use of Action Words significantly decrease the cost of automation and accelerate the overall testing cycle

So your acceptance tests are...

Understandable

Definition of business domain tests enable better alignment of distributed team

Maintainable

Refactoring and optimization features dramatically accelerate maintenance



Can be automated

Good design based on Action Words streamlines the test automation phase

Questions & Answers