Introduction to testing scenarios

by Pamela Cespedes - Wednesday, June 08, 2016

http://blog.belatrixsf.com/introduction-to-testing-scenarios/

What do we mean by testing based on a scenario?

Introduction to testing scenarios. Testing an application using scenarios means testing the end-to-end functionality or a feature, in order to assure it works as expected. We are going to check if the possible workflows are working as we want them to. When we test an application using scenarios, we are testing the complete functionality as the end user, for example if you want test an ecommerce website (such as www.buyyourfavoritebook.com), a scenario could be: Buy books from Stephen King. In this example we will test complete functionality, not just small pieces. Before writing the scenario the tester needs to have a deep knowledge of the business requirements, and as a result the tester needs to work in close collaboration with other stakeholders such as the product owner and business analyst.

What is the difference with test cases?

A test case is a set of conditions under which a tester will determine that an application, functionality, or software is working as expected. The test case has preconditions, steps, and an expected result. Test cases focus on testing small pieces of functionality. For example in our previous example (scenario: buy books from Stephen King) we will have several test cases, for example: login into website with a valid user/pass; select the author and choose the books; buy the books. A scenario can have more than one test case. The key difference is:

- **Test Case:** “How to be tested?”
- **Test Scenario:** “What to be tested?”

When can we test using scenarios?

I recommend using scenarios in the following situations:

- **Business rules are complicated.** It is easier to think about situations like the end user (real situations, usual workflows) instead of following steps that check only a small piece of the system. This allows testers to see the big picture and then go into details (requirements), in order to be able to write scenarios while thinking as an end user.

- **When the systems change rapidly. Creating and maintaining test cases involves time. If we are working on an application where we needed the changes yesterday, we cannot spent time creating and maintaining test cases for each small piece of the system. A scenario can be written in one line.**

- **When we have a large number of data combinations or possible paths.** In this situation we need to prioritize the testing efforts, because we cannot spent time testing all possible

combinations. We need to focus on the most common scenarios, workflows, and transactions.

- **Where test cases cannot be reused.** As we said before, test cases are atomic, because we test small pieces of an application in a test case. For example, if we created test cases for an insurance company, and our website allows users to buy homeowner policies in New York (where the state allows the customers to have just one policy for more than one house), but then we start working on California (where the state says a customer should have a separate policy for each house). Here we cannot re-use our test cases for different states. We will need to write a different set of test cases for each state. Also, if there is not another state that has the same rules as New York, then we can't re-use our test cases. And if the New York state changes its laws, for example to say you can only have a homeowner policy if you have life insurance, then we will need to update all our test cases.

5 top tips when creating testing scenarios

Based on my experience, here are my top tips:

1. Properly understand the business rules.
2. Discuss with the business analyst and product owner to get more business details.
3. Identify the most common workflows.
4. Think as an end user, and follow the final user workflow. This will help you to think in scenarios.
5. Identify the final result of a workflow. It's important to understand why we are adding/updating an application.

3 top testing tips

1. Focus on the final result.
2. Test first the happy path, and don't get lost in complicated scenarios. We need to identify ASAP if something is not working as expected.
3. List all the possible scenarios and prioritize.
