~ Unit 2. Test Cases and Test Suites ~

Test Cases / Part 1

First we make our habits, then our habits make us.
- Charles C. Noble

Quick Intro

Before going on a fishing trip, Mr. Wilson came up with the following list:

1. Fishing pole
2. Box with extra hooks, fishing lines, and other gear
3. Can of worms
4. Beer mug
5. Bottle of beer
6. Potato chips

In the morning Mr. Wilson takes that list and checks his backpack for the availability of each of the 6 objects.

Each item on his list is a test case.
The list itself is a test suite.
The process of figuring out and writing down each item is test case generation.
The process of checking the backpack for the availability of each object on the list is test case execution.

An essential part of the test case is the expected result—for instance, "Bottle of beer". Thus, a test case can consist of only an expected result.

Please, note that in this **Unit** we'll talk about **formal side** of a test case; in other words, test case structure, useful attributes, formatting, etc. In future lectures, we'll learn about the **actual side** of a test case; that's about test case content that would have high probability of finding a bug.

Test Case Structure

The problem with a test case having **only the expected result** is that the expected result alone might not be sufficient to perform the test case execution. In the case of a bottle of beer, we can just unzip the backpack and look inside. In the case of software, as a rule, we need **an instruction** on how to reach an actual result.

Our red-hot startup, www.sharelane.com, has a new employee; let's call him Samuel Pickwick.

Some friendly fellow hands him the following one-line test case:

"Payment can be made by Visa."

Mr. Pickwick's brain immediately generates two perfectly reasonable questions:

- "How can I do testing if I don't have a Visa card and don't know where to get one?"
- "How can I be sure that a payment was really made even if I had that Visa card?"

One thing that is more or less clear is the purchase process (for example, create new account and log in, find a book to buy, put it into the shopping cart, do checkout), but even with this common knowledge, the test case execution will be stuck at the point where the tester enters credit card information.

In other words, that test case cannot be executed by Mr. Pickwick without someone helping him.
This situation is bad because:

- It adds even more stress to Mr. Pickwick's first days at his new company.
- Mr. Pickwick MUST interrupt the work of other testers to get the information he needs.

It's completely different situation when Mr. Pickwick is given a proper test case.

Our red-hot start-up, www.sharelane.com, has a new employee; let's call him Samuel Pickwick.

Some friendly fellow hands him the following test case:

SL

1. Go to http://main.sharelane.com.
2. Click link "Test Portal".
3. Click link "Account Creator".
4. Press button "Create new user account".
5. Copy email to the clipboard.
6. Go to http://main.sharelane.com.
7. Paste user email into textbox "Email".

8. Enter "1111" into textbox "Password".
9. Press button "Login".
10. Enter “expectations” into textbox "Search".
11. Press button "Search".
12. Press button "Add to Cart".
13. Click link "Test Portal".
14. Click link "Credit Card Generator".
15. Select "Visa" from drop-down menu.
16. Press button "Generate Credit Card".
17. Copy card number to the clipboard.
18. Go to http://main.sharelane.com.
19. Click link "Shopping cart".
20. Press button "Proceed to Checkout".
21. Select "Visa" from drop down menu "Card Type".
22. Paste card number into text box "Card Number".
23. Press button "Make Payment".
24. Write down order id: ___.
25. Click link "Test Portal".
26. Click link "DB Connect Utility".
27. Make database query:
    select result from cc_transactions where order_id = <order id>;

Expected result is **10**

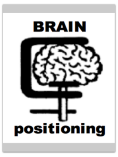
In our last example (let's refer to it as **Test Case with Credit Card**), the Expected Result was preceded by steps that are supposed to lead the test case executor to an Actual Result. These steps are called the **procedure.**

Here is an analogy for you:

- The procedure is a set of steps on a staircase.

- The expected result is a certain object we are supposed to find if we climb to the top of that staircase.

- The actual result is what we actually find when we reach the top of the staircase.



If we apply the computer science concept of input/output, we'll have this:

- The **procedure** is the *instruction about input.*

- The **execution of the procedure** is the *act of input.*

- The **expected result** is the *expected output.*

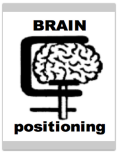
- The **actual result** is the *actual output.*

Results of the Test Case Execution

The test case execution is complete once we have compared the actual and expected results. This comparison can give us one of two results:

1. **PASS –** that's when Actual Result equals Expected Result

2. **FAIL –** that's when Actual Result does not equal Expected Result. Is that necessarily a bug? Let's see.



If the test case execution result is FAIL it doesn't necessarily mean that we found a bug.

Here is typical start-up situation: product manager communicated certain change with programmer, but none of them communicated that change with test engineer.

So, code is going to be **legitimately** changed, but the test engineer is not aware of it and test case execution will end up with FAIL. Thus the FAIL would be caused not by bug, but by the fact that expected result **known to tester** is wrong.

Sometimes we have a situation where the test case execution is blocked, so we cannot get to the actual result.

For example, in Test Case with Credit Card, we can be stuck at **Step 23** if the *Make Payment* button doesn't exist on the Web page. In that case, we would file a bug about the absence of the *Make Payment* button, and we would delay the execution of this test case until that bug is fixed.

Useful Attributes of the Test Case

In addition to essential parts like expected result and procedure, test case can have a number of useful attributes to make our lives easier:

Unique ID
Priority
IDEA
SETUP and ADDITIONAL INFO
Revision History

UNIQUE ID

This is an extremely important attribute. A test case without an ID is like a house without an address. The test case ID should be unique, not only within the concrete test suite which contains that test case, but also within all test suites inside the company. The rationale for this is that, as time goes by, it will be necessary to keep test case statistics; update, remove, or move test cases between test suites, etc.

PRIORITY

The test case priority reflects how important this test case is. The priority is graded from 1 to 4, with 1 being the highest priority.

For example, the test case that checks if the Make Payment button works must be P1, and the test case that checks the color of the Web link "Contact" is P4.

Why do we need to prioritize test cases? Let's assume that we have two test cases, P1 and P3. We have time enough to execute only one of them. Which one do we pick? P1, of course!

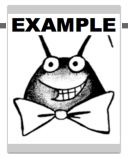
Test case prioritization is especially important during regression testing, that's testing of old functionalities. Later on we'll talk about regression testing a lot.

Question: How do we assign a test case priority?
Answer: As a rule, the author of the test case decides the importance of the thing being checked by that test case.

IDEA

This is a description of **what** we check for in a particular test case.

In the Test Case with Credit Card, Expected Result is **10**. What info do we get by looking at **10**? Nothing except two digits. However, the developers of



www.sharelane.com have developed the following process to create a code for the result of a transaction:

- The **first digit** corresponds to the internal id of the credit card: **1** for Visa, **2** for MasterCard, and **3** for AmEx.

SL See database table cc_types: Test Portal>DB>Data>cc_types.

- The **second digit** is the success code: **0** for a successful transaction, and **1** for a failed transaction.

Now we have the full understanding what **10** means: the transaction with Visa was successful.

The problem is that **even the author** of the Test Case with Credit Card can forget what **10** means. That's why at the beginning of a test case its author should put in human language the following phrase: "Payment can be made by Visa," so everyone who must execute that test case will immediately understand what is being tested.

SETUP AND ADDITIONAL INFO

A culinary recipe consists of two parts:

1. A list of ingredients
2. Instructions on how to fry, steam, or bake those things.

The first part of a recipe is needed to enable a chef to

- see beforehand
AND
- in one place

all the necessary ingredients of the recipe and have them ready once the time comes to prepare it.

This is a very practical first step.

The setup part of a test case can include:

- Information about existing (also called legacy) user accounts or instruction on how to create a new user account.

- Data used in the test case; for example, default password for testing.

- SQL (*pronounced "sequel"***)** queries (also called "*SQL statements*") are the commands used to interact with the database (short for database is DB). *SQL stands for Structured Query Language.*

- Comments to help the tester; for example, instructions about where to get the software needed for the test case execution (for example, Windows software to connect to the DB).

- Other items that can ease the execution and maintainability of the test case (we'll talk about maintainability in a minute).

REVISION HISTORY

In order to have data about the birth and history of the modifications for each test case, we create a mini-journal of any changes where we record: When, Who, Why and What.

We'll have these statuses:

- Created (date/name) for when the first version of this test case was created and who created it.

- Modified (date/name) for when any change to the test case took place and who was responsible for the change

- Reason – why the test case was changed and what was changed

Test case management systems record revision history automatically.

Now, let's create the **Test Case with Credit Card** using all mentioned test case attributes.

<table>
<tr><td colspan="2">IDEA: Payment can be made by Visa</td><td>TC ID</td><td>CCPG0001</td></tr>
<tr><td colspan="2"></td><td>Priority</td><td>1</td></tr>
<tr><td colspan="4">SETUP and ADDITIONAL INFO

Password: "1111" (this is default unchangeable password for all user accounts)
Search keyword: "expectations"

!!!Go to Test Portal>Helpers>DB Connect Utility to run SQL query below.

SQL1:
select result from cc_transactions where order_id = <order id>;</td></tr>
<tr><td>Created (date/name):
11/17/2004/O.Ferguson</td><td colspan="3">Reason: new way to verify that credit card transaction was successful.</td></tr>
</table>

Modified (date/name):	Reason:
SL 1. Go to http://main.sharelane.com. 2. Click link "Test Portal". 3. Click link "Account Creator". 4. Press button "Create new user account". 5. Copy email to the clipboard. 6. Go to http://main.sharelane.com. 7. Paste user email into textbox "Email". 8. Enter **password** into textbox "Password". 9. Press button "Login". 10. Enter **search keyword** into textbox "Search". 11. Press button "Search". 12. Press button "Add to Cart". 13. Click link "Test Portal". 14. Click link "Credit Card Generator". 15. Select needed card from drop-down menu. 16. Press button "Generate Credit Card". 17. Copy card number to the clipboard. 18. Go to http://main.sharelane.com. 19. Click link "Shopping cart". 20. Press button "Proceed to Checkout". 21. Select appropriate value from drop down menu "Card Type". 22. Paste card number into text box "Card Number". 23. Press button "Make Payment". 24. Write down order id: ___. 25. Run **SQL1**	✓ ***10***

Data-Driven Test Cases

The main advantage of our brand-new Test Case with Credit Card is that we don't have to modify the steps necessary to test other credit cards if we want to test them in a similar way. One thing we do need to modify, however, is the test case **DATA**. So, if we want to test two more cards the same way we just tested Visa, we

- Do "copy" one time

- Do “paste” two times
- In both new test cases, we don't change any steps. Instead, we change only the two values located in the header and the expected result (remember that we also have to change the test case ID and any other attributes). These values are Visa and 10.

This type of test case is called **data-driven**, because the **data and the steps using that data are not mixed up, but separated and linked to each other.**

Lesson Recap

1. An essential part of the test case is the expected result.

2. The test case steps (procedure) serve as an instruction of how to reach an actual result (output).

3. The test case execution ends with a PASS or FAIL result. A FAIL result is desired, because in that case we have a bug (provided that the test case reflects what's really expected).

4. The test case execution is considered incomplete if the tester couldn't execute **all** the steps: for example in a situation where the test case execution is blocked because of a bug in the middle of the execution.

5. The following test case attributes are very useful:

- A **unique ID** is unique not only within a test suite, but also among **all** test cases existing in the company.

- **A priority** is a value from 1 to 4 (inclusively) which reflects the importance of a test case. The practical consequence of having differences in priorities: A *P1 test case is more important than a P2 test case, and if we have time to execute only one of the two, we would execute the P1 test case.*

- An **IDEA** – a written description (in human language) of a test case idea.

- The **SETUP AND ADDITIONAL INFO** gives the test case executor helpful information for test case execution. This section is commonly used to store data, account information, SQL statements, and other pieces of information which help to execute test cases and make them more maintainable.

- The **revision history** helps us to know history behind test case modifications.

6. In **data-driven** test case**,** the data and the steps using that data are not mixed up, but separated and linked to each other.

Homework

1. Run (execute) a test case with Visa.

2. Get used to the interface of ShareLane and Test Portal. Play with it, click links, make test purchases, etc. Remember to only use a credit card from Credit Card Generator.

NEVER use actual credit card or other actual personal/financial data during interactions with ShareLane or Test Portal.

3. Create and run test case with MasterCard. Expected result after your query database is **20**. What actual result do you get? (Hint: you are going to see a bug).

Quiz

Question 1: Check one essential attribute of test case.

- ☐ Steps
- ☐ Expected result
- ☐ Actual result
- ☐ Setup

--

Question 2: If test case execution results in FAIL, it means we have a bug.

- ☐ True
- ☐ False

--

Question 3: Test case priority is needed

- ☐ for situations when we are short on time and need to execute only important test cases
- ☐ to mark the most important test scenarios
- ☐ All of the above

--

Question 4: Test case procedure is a sequence of steps that lead to an actual result.

- ☐ True
- ☐ False

--

Question 5: Setup and Additional info section of a test case is needed to specify expected result.

- ☐ True
- ☐ False

--

Question 6: SQL is...

- ☐ a test case management software.
- ☐ a software to generate credit cards for testing.
- ☐ A language to communicate with a database.

--

Question 7: It's OK to have two or more test cases with the same ID.

- ☐ True
- ☐ False

--

Question 8: It's OK to change Test Case ID.

- ☐ True
- ☐ False