All About Requirements

How to succeed with business oriented IT projects using business process analysis, use cases and requirements management in general.

15/10/2011

Test Cases Based on Use Cases

Use Cases represents functional requirements and functional requirements need to be tested.

The purpose of the functional test is:

A) Determine that the delivered solution works as required

B) Determine that the delivered solution works as specified

The difference between A) and B) is that in A) you test that the requirements you have written in the use case (being the flow, the pre-conditions, the post-conditions, and the business rules) are actually delivered and works as required. In B) however, you test that the delivered solution works as specified in the design document.

For example if you have required that it must be possible to create users in the system (specified in the use case "Create User"). That use case specifies the functionality and the conditions under which "create user" must work.

What the use case (usually) does not say is how the requirement is designed in the system – for example that the create user functionality is designed as a wizard functionality with three tabs and with all fields popping up when required. That part, the design needs testing as well, so that you determine that not only is the functional requirement delivered but it also behaves as specified in the design specification.

In this post I will outline how to create a full functional test case foundation on basis of your use cases. Normally you would need to also update the functional test cases to address the design specification. In addition, besides functional testing various other test types exist. However, in this post I will focus only on the functional testing based on use cases. All the examples will be based on the use case example "Create User" Download Use Case Example - Create User

The nature of the use case is that it describes the steps of activities inside the use case and for each step a number of business rules are related. So, in theory the use case could be considered a test case because you just need to test that you can execute the use case as specified and get the results as expected. In other words you test that:

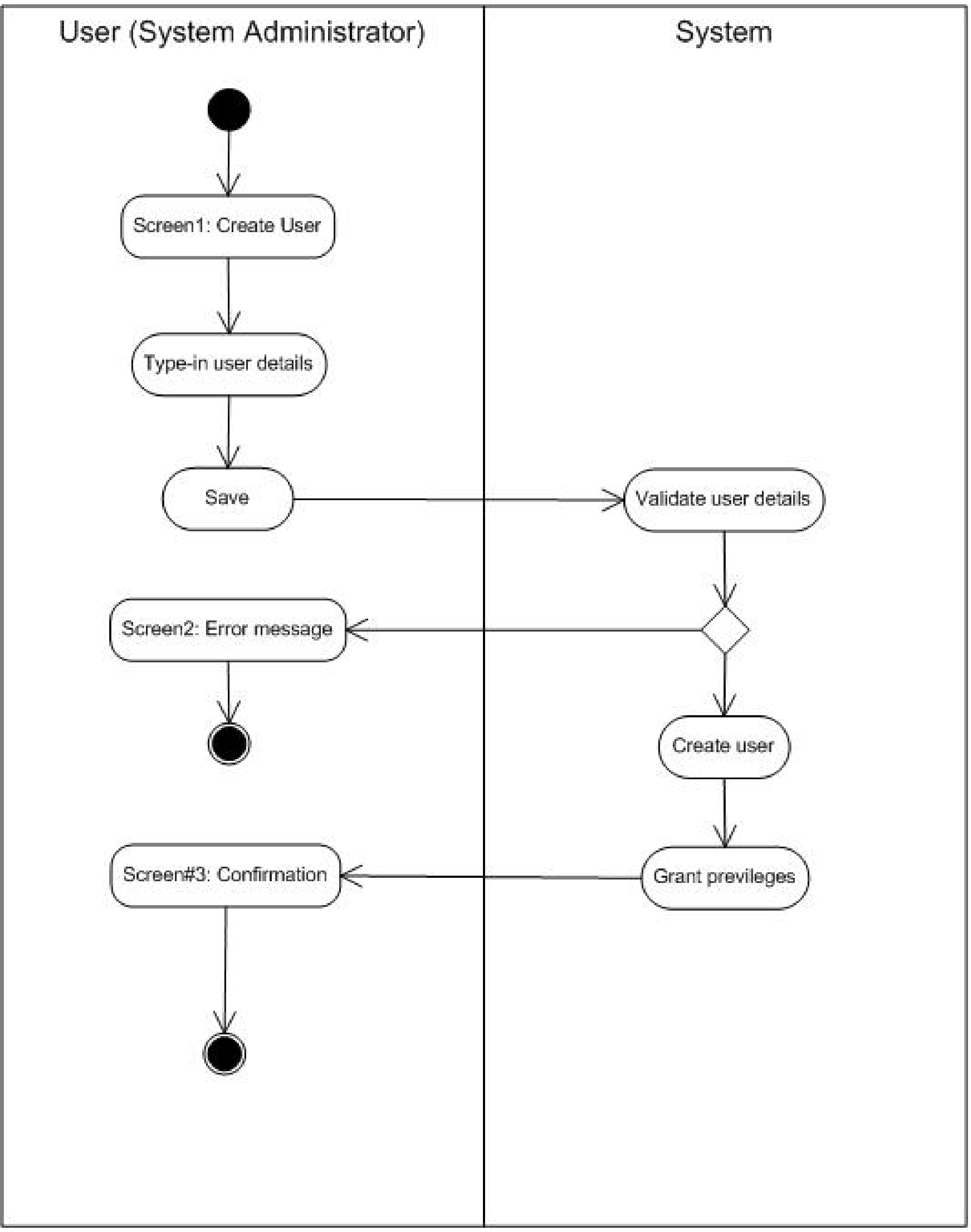
A) The use case can not be initiated if any of the pre-conditions are broken

B) The use case can be executed following the steps outlined in the use case

C) The use case address all business rules outlined related to the right steps

D) The use case produce the conditions and / or output as specified in the post-conditions

In B) it says, "The use case can be executed following the steps outlined in the use case". If we take a look on the steps in the use case (illustrated graphically in the activity diagram):



As you can see the steps through the use case is:

Step	*User (System Administrator)*	*System*
10	*Screen1: Create User*	

20	*Type-in user details*	
30	*Save*	
40		*Validate user details*
50		*Create user*
60		*Grant privileges*
70	*Screen3: Confirmation*	

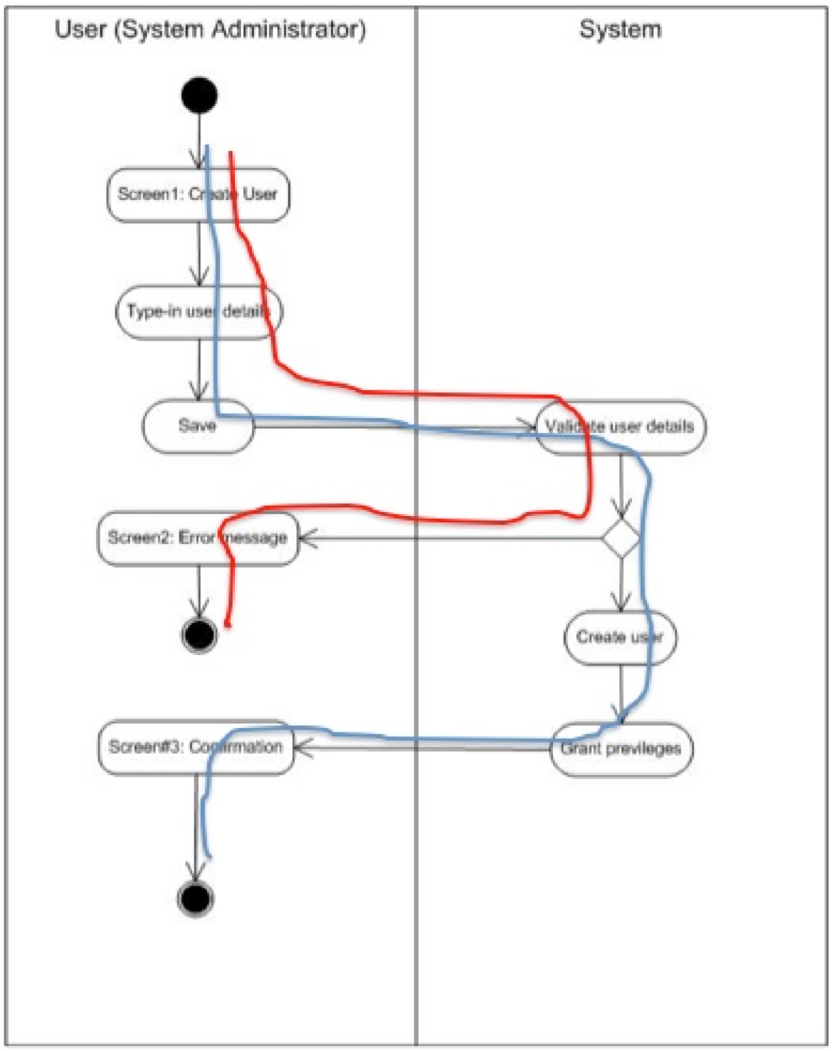
But the use case could also be executed through these steps:

Step	*User (System Administrator)*	*System*
10	*Screen1: Create User*	
20	*Type-in user details*	
30	*Save*	
40		*Validate user details fail*
50	*Screen2 Error message*	

This lead to rule of thumb #1:

For each "way" through a use case you need a test case

In this example we need two test cases to represent the use case test. This can also be illustrated as the picture below. Here you can see a coloured line for each "way" through the use case, each representing a test case.



All right, so far we have identified two test cases for the use case test. Each test case has its own flow (the steps). Lets take a look at one of the test cases (for simplicity each time the test case is outlined it is written in **green**):

Step	*User (System Administrator)*	*System*
10	*Screen1: Create User*	
20	*Type-in user details*	
30	*Save*	
70	*Screen3: Confirmation*	

Actually, when executing a test case what the tester needs to do is to perform the user steps. Therefore I have deleted the system-side steps because the tester does not execute these steps – the system-side steps are not tested, but the outcome is. In other words the tester performs her steps and verifies that the system has responded correctly.

In order to verify that the system has responded correctly you need post-conditions in your test case. The post-conditions can be copied from the use case.

In the use case the following post-conditions were specified (for the positive flow, which we are handling here):

If success:

- A new user is created in the system
- The user created is granted the privileges as specified
- The user created is created with a unique 4 digit employee number and a default password

Each of the bullets above represents a post-condition that needs to be verified. ***Your test case now looks as specified below:***

Step	*User (System Administrator)*	*System*
10	*Screen1: Create User*	
20	*Type-in user details*	
30	*Save*	
70	*Screen3: Confirmation*	

Post-conditions:

1. It must be verified that a new user is created in the system with the attributes typed-in
2. It must be verified that the user created has been granted exactly the privileges as specified
3. It must be verified that the user created is assigned a 4 digit employee ID
4. It must be verified that the user created is assigned to a password

As you can see the post-conditions outlined is following the structure "verify that…". Depending on the systems nature and the testers skills you could outline ***how*** to verify the statements outlined in the post-conditions. However, in this example we'll assume that the testers are capable of verifying the statements.

Now we have the steps the tester must perform and the post-conditions the tester needs to check-for or verify. If we take a look at the steps you can see that the steps are not very clear. For example in step 20 it says "Type-in user details". But what user details?

According to the use case, when typing-in user details the following business rules apply:

FR1	**User details**	**It must be possible to specify the following details when creating a new user:**

		• **First name** • **Surname** • **Birth date** • **Address** • **City** • **Zip code** • **Telephone** • **Gender** • **email**
FR2	Specify related department	It must be possible to specify that the user to be created belongs to one of the following departments: • Marketing • Sales • R&D • IT • Administration
FR3	Permissions	It must be possible to specify the following permissions to the system: • Read only • Read/write

A good test case specifies in details what data to type-in in the various steps. The reason is that you need to be able to reproduce an error by using the exact same data and you need to be able to re-test and confirm error correction by executing the test case with the exact same data again. This lead to:

Rule of thumb #2:

Test cases must be able to be re-produced by using a specified set of (test) data.

You could choose to refer to an excel sheet with the test data outlined – especially if you plan on reusing the test data for other test cases. For simplicity I will specify the test data directly in the test case.

Your test case now looks as specified below:

Step	*User (System Administrator)*
10	*Screen1: Create User*
20	*Type-in user details:* • First name = “Jan” • Surname = “Thomson” • Birth date = “12-12-1970” • Address = “122, Main Road”

- City = “Manchester”
- Zip code =XX 555”
- Telephone = “999 999”
- Gender = “Male”
- Email = jan@com.com

- Department = “Sales”
- Permission = “Read/Write”

30 *Save*

70 *Screen3: Confirmation*

Post-conditions:

1. It must be verified that a new user is created in the system with the attributes typed-in ***in step 20***
2. It must be verified that the user created has been granted exactly the privileges as specified ***in step 20***
3. It must be verified that the user created is assigned a 4 digit employee ID
4. **4.** It must be verified that the user created is assigned to a password **being 8 character long as specified in business rule “FR 7”**

The test case could be complete by adding the pre-condition, a unique test case ID and a test case name.

Your test case now looks as specified below:

Test case ID: TC #1

Test Case Name: Create User_Main flow

Pre-condition:

- The user must be logged-in as “System Administrator”

Flow:

Step *User (System Administrator)*

10 *Screen1: Create User*

20 *Type-in user details:*

- First name = “Jan”
- Surname = “Thomson”

- Birth date = “12-12-197 0”
- Address = “122, Main Road”
- City = “Manchester”
- Zip code =XX 555”
- Telephone = “999 999”
- Gender = “Male”
- Email = jan@com.com

- Department = “Sales”
- Permission = “Read/Write”

30 Save

70 Screen3: Confirmation

Post-conditions:

1. It must be verified that a new user is created in the system with the attributes typed-in in step 20
2. It must be verified that the user created has been granted exactly the privileges as specified in step 20
3. It must be verified that the user created is assigned a 4 digit employee ID
4. It must be verified that the user created is assigned to a password being 8 character long as specified in business rule “FR 7 ”

So, now you have created the first of two test cases related to the use case “Create User”. By executing the test case just created are you absolutely confident that “Create User” works? Well, definitely for the test data typed-in we are – but what if the user to be created should be related to the “Marketing” department instead, will it still work? The case is that you can’t de 100% sure unless you test that too. What we are talking about is test variations.

This lead to rule of thumb #3:

Only by testing all variations of the test case you can be sure that the functionality works error free.

Test variations are the combinations of choices you can type-in. The choices are hidden in the business rules. For example, by looking at the use case and business rule “FR 2” you could execute the test case just created with the related department being Marketing – or Sales or R&D or IT or Administration….and then again, you could try granting the permission “Read only” (FR 3). Notice that the variations of the business rules “FR 4” and “FR 5 “ belongs to the other test case not outlined in this example (testing for the error situation).

Despite the variations of choices you could also create different test data and try those with different test variations. For example in the test case example just created we type in the date 12-12-2011 – maybe it could be interesting typing-in other dates too. Or try creating a female user etc.

The number of test variations can be huge and you need to determine what level is right for you depending on the system nature and how business critical it is – I will address this in another post.

Email this

Posted at 11:46 PM in Test, Use Cases | Permalink ShareThis
| Reblog (0)

Comments

You can follow this conversation by subscribing to the comment feed for this post.

Verify your Comment

Previewing your Comment

Posted by: |
This is only a preview. Your comment has not yet been posted.

Post Edit

Your comment could not be posted. Error type:

Your comment has been posted. Post another comment

The letters and numbers you entered did not match the image. Please try again.

As a final step before posting your comment, enter the letters and numbers you see in the image below. This prevents automated programs from posting comments.

Having trouble reading this image? View an alternate.

Continue

All About Requirements