Reducing test case documentation time

Author: Ranjit Shewale
jcrvs@hotmail.com

Summary

This article describes about the approach to reduce documentation and correction overhead of test cases for the test team so they can spend more time on creating innovative test cases and actual testing. This also leads to a structured documentation approach of test case during the test case preparation phase.

Ask yourself

1. How much time does your team spend in documenting test case?
2. How much time do you spend in internal reviews?
3. How much time do you spend in correcting it?
4. How much time do you spend in correcting the test cases if the client sends across the review comments prior to signing it off?

If the answer to above is 'most of the time', then you need a structured approach to test documentation.

Why would you want to save the time?

As tester you always have to cope with the project pressures as to learn requirements in short time, to document test cases and to execute the cases. Not to forget there are quality overheads, timesheets, project confusions, some re-work here and there. When you are coping with these activities, do you have time to automate them? Do you have time to version control the code? (Test engineers are good version control guys as well), Do you have time for exploring new test techniques for testing and implementing them.

With low time to market in the current web based e-commerce projects how do you cope with short time? Are there any techniques to save time so it can be used effectively elsewhere?

If more time is available the testers can evaluate new tools and techniques to test the products even better. They can study the existing implemented system to design cases in more depth.

How do I save time during the test preparation phase?

Consider the test life cycle. As per the V-model, it has phases as

1. Requirements review, acceptance plan and '***acceptance test case documentation***' during Requirement phase of SDLC,
2. Test planning and '***system test case documentation***' during the analysis phase of SDLC,
3. Integration test plan and '***test case documentation***' during the design phase of SDLC,
4. '***Documenting unit test cases***' and executing them during the coding phase of SDLC,
5. Integration testing during the Project Integration phase of SDLC,
6. System testing during after the integration is complete of SDLC,
7. Bundling and Installation test during code delivery phase of SDLC,
8. Acceptance testing during acceptance at client of SDLC.

The 50% of the tester's time is spent effectively in test case documentation. Not to mention tester spends more time re-documenting the test cases when a change request is encountered during the above phases. More correction time is needed when the project nears completion. This, at times, may even eat up the test execution time.

The solution is to implement templates effectively for documentation of the test cases. The next question that pops into user's mind is 'How do I implement the templates and how will it reduce time?'

Try the following steps: -

Step 1: Do the ground work

1. The review comments (internal peer reviews and external client review comments) are a good source of the information.
2. Visit the review comment document repository and classify the data under the common sections as 'deviation from functionality', 'functionality not comprehended', 'spell mistake', 'bad grammar' etc
3. Study the approach of test case documentation. Do you create test suites to group test cases? Can you club Integration test cases with partially unit and partially system test cases (although this is not a good approach but if acceptable to client then fine)?
4. Is client more specific of the documentation style? As the use of 'American-English', grammar etc

Step 2: Documentation templates for environment familiarity

Now when the groundwork is done, the next step is creating a documentation template for environment familiarity. A familiar look of the document gives more comfort to the client or reviewer since he is used to the document layout and places where to seek information.

It is recommended to create template per test suite. In layman's terms, a separate template for unit test documentation and a separate template for system case documentation.

This reduces the reviewer's time to get familiar with the document format and directly get to the review work.

Refer the following links for few templates: -

1. For unit test: http://www.stickyminds.com/sitewide.asp?sid=2019124&sqry=%2AJ%28MIXED%29%2AR%28createdate%29%2AK%28simplesite%29%2AF%28shewale%29%2A&sidx=5&sopp=10&ObjectId=6198&Function=DETAILBROWSE&ObjectType=TEM
2. For system test: http://www.stickyminds.com/sitewide.asp?sid=2019124&sqry=%2AJ%28MIXED%29%2AR%28createdate%29%2AK%28simplesite%29%2AF%28shewale%29%2A&sidx=4&sopp=10&ObjectId=6197&Function=DETAILBROWSE&ObjectType=TEM

Step 3: Templates for documenting test cases themselves

Approach to document in a familiar format leads to time saving, how:

1. Applying the same principle of the familiarity, if each test case itself is written in a familiar format the reviewer time to get familiar with the documentation style is reduced,
2. The pre-existing test case template itself can be used to 'copy-past-edit' creation of the test cases, thus most of the time is reduced for the tester who writes the cases.

For example consider the following test case.
Steps: 'Check the **User name** text control'
Expected results: 'The data should be fetched from the **Login** column of the **User** table in the database and displayed.'
(*Note*: There are other informational attributes to documenting test case but our documentation style considers only above two).

Such cases can be documented into a template with the bold character replaced with parameter that needs to be edited later. Scrutinize each test case and reframe the description to move the test case to the template. Eventually the template will be full of the generic test cases that can be used across all projects

Not all test cases can be documented in such manner. It may be impossible to generalize the system test cases into the template in such manner as they are really diverse and are application specific.

There can be a catch in this approach: The 'copy-past-edit' test case creation approach reduces the documentation efforts to a large extent. But caution needs to be exercised as it may lead to increases review and re-work if not implemented religiously.

The time management is one of the critical things in project management, as many techniques as possible should be used to generalize test items and implemented to save the test time for the sake of the quality of product.