**Live Project Bug Tracking, Test Metrics, and Test Sign off – Free QA Training Day 6**

Posted In | [Bug Defect tracking](http://www.softwaretestinghelp.com/category/bug-defect-tracking/), [Software Testing Templates](http://www.softwaretestinghelp.com/category/software-testing-templates-test-cases/), [Software Testing Training](http://www.softwaretestinghelp.com/category/software-testing-training/), [Testing Life cycle](http://www.softwaretestinghelp.com/category/testing-life-cycle/)

**This is the concluding part of our “[software testing training on a live project](http://www.softwaretestinghelp.com/free-online-software-testing-qa-training-course/" \o "QA training)” series.**

It is going to be about defects and also few remaining topics that will mark the completion of Test execution phase of the STLC.

In the [previous article](http://www.softwaretestinghelp.com/test-execution-software-testing-qa-training-on-a-live-project-day-5/" \o "Test execution), while Test Execution was going on, we encountered a situation where the expected result of test case was not met. Also, we identified some unexpected behavior during exploratory testing.

**What happens when we encounter these deviations?**

We obviously have to record them and track them to make sure that these deviations get handled and eventually fixed on the AUT.

**#1.** These deviations are referred to as Defects/bugs/issues/incidents/errors/faults.

**#2.** All the following cases can be logged as defects

* Missing requirements
* Incorrectly working requirements
* Extra requirements
* Reference document inconsistencies
* Environment related issues
* Enhancement suggestions

**#3.** Defect recording is mostly done in excel sheets or via the use of a defect management software/tool. For information on how to handle defects via tools, try using the following links:

* [HP ALM](http://www.softwaretestinghelp.com/hp-quality-center-tutorial-6/" \o "Defect recording in HP ALM)
* [Atlassian JIRA](http://www.softwaretestinghelp.com/atlassian-jira-tutorial-2/" \o "Defect tracking in JIRA)
* Also, refer this post for a list of the [most popular bug tracking tools](http://www.softwaretestinghelp.com/popular-bug-tracking-software/" \o "Most popular bug tracking tools) in the market.

**How to Log the Defects Effectively:**

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/defect-report-and-test-sign-off-2.jpg)

We will now try to see how to log the defects we encountered in the previous article in an excel sheet. As always, choosing a standard format or template is important.

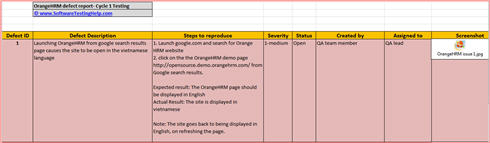
**Typically, the following columns are a part of defect report:**

1. **Defect ID:** for unique identification.
2. **Defect Description:** This is like a title to describe the issue briefly.
3. **Module/section of the AUT:** this is optional, just to add more clarity as to indicate the area of the AUT where the problem was encountered.
4. **Steps to reproduce:** What is the exact sequence of operations to be performed on the AUT to recreate the bug are to be listed here. Also, if any input data is specific to the problem that information is to be entered as well.
5. **Severity:**  To indicate the intensity of the issue and eventually the impact this might have on the functioning of the AUT. The guidelines on how to assign and what values to assign in this field can be found in the test plan document. So, please refer to the [test plan document from article 3](http://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/" \o "Test plan writing process).
6. **Status –** will be discussed further in the article.
7. **Screenshot:** A snap shot of the application to show the error when it happened.

These are some of the ‘must-have’ fields. This template can be expanded (E.g. to include the name of the tester who reported the issue) or contracted (E.g. the module name removed) as needed.

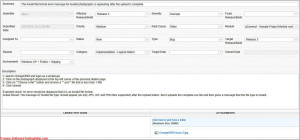
Following the above guidelines and using the template above, a sample Defect log/report could look like this:

**Sample Defect Report for OrangeHRM Live project:**

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Defect-Report-OrangeHRM-SoftwareTestingHelp.xlsx)

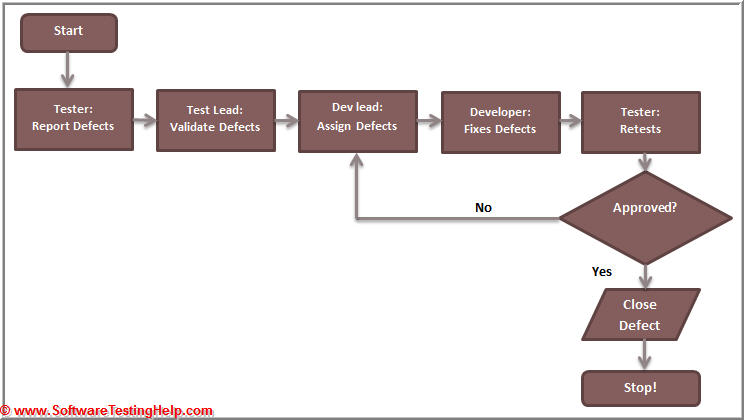
**=> [Click here to download live project Defect report](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Defect-Report-OrangeHRM-SoftwareTestingHelp.xlsx" \o "Samle defect report live project" \t "_blank)**

**Below is the sample defect report created in qTest test management tool: *(Click on image to enlarge)***

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/defect-report-sample.jpg)

Defects are no good when we log them and keep them to ourselves. We will have to assign them in a right order to have the concerned teams act on them. The process – who to assign or what order to follow can also be found in the test plan document. It is mostly similar to: *(Click on image to enlarge)*

**Defect Cycle:**

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Defect-life-cycle-1.jpg)

From the above process it can be noted that, bugs go through different people and different decisions in the process of being identified to fixed. To track and to establish a transparency as to exact what state a certain bug is at, a “Status” field is used in the bug report. The entire process is referred to as a “Bug life cycle”. For more information on all the statuses and their meaning, please refer to this [bug life cycle tutorial](http://www.softwaretestinghelp.com/bug-life-cycle/" \o "Bug life cycle).

**A few pointers while Bug Tracking:**

* When we are new to a create team/project/AUT, it is always best to **discuss the issue we encountered** with a peer to make sure that our understanding on what really makes for a defect is correct or not.
* Do **provide all the information** that is necessary to reproduce the issue. A defect that comes back to a testing team with the status set as “Not enough information” does not reflect very positively on us. Check out this post – [How to get your all bugs resolved without any ‘Invalid bug’ label](http://www.softwaretestinghelp.com/how-to-get-your-all-bugs-resolved/" \o "Invalid bug).
* Check if a similar issue was raised before creating a new one.**‘Duplicate’ issues** are also bad news for the QA team.
* If there is an issue, that comes up randomly and we don’t know the exact steps/situations in which we can recreate the issue- raise the issue all the same. At the risk of the issue being set to **“Irreproducible/not enough information”** – we still have to make sure that we handled all possible malfunctions to the best extent possible.
* **The general practice** is that the QA team creates each one’s defects in an excel sheet during a day and consolidates it at the end of the day.

**The Complete Defect Life Cycle:**

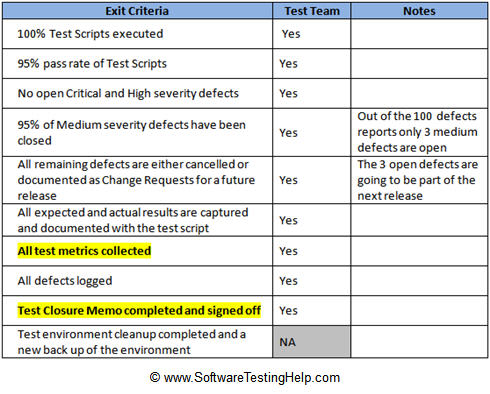
**For our live project if we were to follow the defect life cycle for defect 1,**

------------

* When I (the tester) create it, its status is **“New”.** When I assign it to the QA team lead, the status is still “New” but the owner is now, the QA lead.
* The QA lead will review the issue and on determining that it is a valid issue, the issue is assigned to the Dev lead. At this phase, the status is **“Assigned”** and the owner is Dev lead.
* The dev lead will then assign this issue to a developer who will work on fixing this issue. The status will now be **“Work in Progress”** (or something similar to that effect), the owner is the developer.
* For defect 1, the developer is not able to reproduce the error, so he assigns it back to the QA team and set the status as**“Not able to reproduce”.**
* Alternately, if the developer was able to work on it and fix issue, the status would be set to **“resolved”** and the issue would be assigned back to the QA team.
* QA team will then pick it up, retest the issue and if it is fixed, will set the status to **“Closed”**. If the issue still exists, the status is set to **“Reopen”** and the process continues.
* Depending on the other situations, the status can be set as**“Deferred”**, “Not enough information”, **“Duplicate”**,**“working as intended”**, etc by the developer.
* This method of recording the defects, reporting and assigning them, managing them is one of the major activities performed by the QA team members during the test execution phase. This is done on a daily basis until a particular test cycle is complete.
* Once Cycle 1 is done, the dev team will take a day or two to consolidate all the fixes and rebuild the code into the next version that will be used for the next cycle.
* The same process again continues for cycle 2 as well. At the end of cycle, there is a chance that there might still be some issues “Open” or unfixed in the application.
* At this stage- do we still continue with Cycle 3? If yes, when will we stop testing?

**Exit Criteria for the OrangeHRM Live Project Testing:**

This is where we employ what we would call the “Exit criteria”. This is pre-defined in the Test plan document. It is simply in the form of checklist that will determine whether we conclude the testing after cycle 2 or go for one more cycle. It looks like, the below when filled out taking into consideration some hypothetical answers to the following questions concerning, OrangeHRM project:

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Test-Exit-Criteria.jpg)

When we look carefully at the above checklist, there is metrics and sign off mentioned there that we have not discussed earlier. Let us talk about them now.

**Test Metrics**

We have established that during the test execution phase, reports are sent out to all the other project team members to give a clear idea about **what is happening in the QA execution phase**. This information is important to everyone in order to get a validation about the overall quality of the final product.

Imagine I report that 10 test cases passed or 100 test cases were executed- these numbers are merely raw data and does not give a very good perspective about how things are going on.

Metrics play a vital role is filling this gap. Metrics are in simple words, **intelligent numbers that the testing team collects and maintains**. For example, if I said 90% of the test cases passed, it makes more sense than saying 150 test cases passed. Isn’t it?

There are different kinds of Metrics collected during the test execution phase. What metrics exactly are to be collected and maintained at what periods of time- this information can be found in the test plan document.

**The following are the most commonly collected test metrics for most projects:**

* Pass Percentage of the Test cases
* Defects density
* Critical defects percentage
* Defects, severity wise number

Check out the [status report attached to this article](http://www.softwaretestinghelp.com/test-execution-report/" \o "Status report) to see how these metrics are used.

**Test Sign off /Completion Report**

As we have to notify all the stakeholders that testing has begun, it is also the QA team’s duty to let everyone know that testing has been complete and share the results. So, typically an email is sent from the QA team (usually the team lead/QA manager) giving an indication that QA team has signed off on the product attaching the test results and the list of open/known issues.

**Sample Test Sign off Email:**

***To:****Client, PM, Dev team, DB team, BA, QA team, Environment Team (and anyone else that needs to be included)*  
***Email:****Hello Team,*

*QA team signs off on the OrangeHRM version 3.0 software after the successful completion of the 2 cycles of functional testing the website.*

*The test cases and their execution results are attached to the email. (Or mention the location where they are present. Or if using test management software, provide details regarding the same.)*

*The list of known issues is attached to the email too. (Again, any other references that make sense can be added.)*

*Thanks,*  
*QA team lead.*

***Attachments:****Final Execution Report, Final issue /defect report, Known issues list*

Once the Test sign off email goes out from the QA team, we are officially done with the STLC process. This does not necessarily mark the completion of the “Test” phase of the SDLC. We still have the UAT testing to finish for that to happen. Find [more details about UAT testing here](http://www.softwaretestinghelp.com/successful-user-acceptance-testing/" \o "UAT Testing).

After the UAT is done, the SDLC moves to deploy phase where it goes live and is available to its customers/end users to be consumed.

That’s it!