Cyreath.co.uk
Software Test <> Quality Assurance

Issue Response Considerations

A discussion on responding to developer issue resolution setting

Discussion Document
Prompted by Sitab Bhandari
Authored by Mark Crowther, Vladimir Trushkin, Sanket Paliwal and Shamsuzzaman Sarker.
With thanks to Aayush Kathuria at http://groups.google.com/group/SoftwareQA-Testing

1.0 INTRODUCTION

When the Test Engineer raises a new issue they set a status such as 'Open', 'For Review' or something similarly appropriate for their organisation. This status tells the development team there is a newly entered issue that needs to be reviewed and actioned by them. At this point the issue is generally believed to be a bug by the Test Engineer but hasn't been accepted as such by the development function as the review is needed. Following a review the development team need to set a new status and so progress the issue through the issue life cycle, described through the workflow of the issue management tool.

When the development team set the new status and carry out the appropriate action the Test Engineer will need to respond and it is the considerations around that response that is the focus of this paper.

2.0 ISSUE RESPONSE AND CONSIDERATIONS

As there is no universal agreement on the exact status settings to be used those given cover the typical responses available within an issue management tool and is by no means exhaustive.

2.1 Fixed

Fixed is the simplest of responses from the Test Engineers perspective. When the developer sets the status to FIXED this means the bug has been accepted and a fix has been delivered. The Test Engineer will need to pick up the fix from the appropriate code branch and rerun the Test Case to verify the developer's fix. If the Test Case passes and the fix is in turn accepted as valid and the Bug record can be closed.

The Test Engineer should also try to do some Ad-Hoc testing in the same area since fixing of the bug may introduce many more.

2.2 Invalid

If a bug is marked as invalid then there are two possibilities to consider. One that it-isn't-a-bug, i.e. what the tester believes they saw as an error isn't erroneous behaviour. This would mean the Test Case could be invalid so the test engineer should check the Test Case and:

- Reflect on why the analysis technique used to derive the Test Case didn't produce a valid case.
- Consider what forms of review and sign off happened at test specification and design and why they were ineffectual to some degree.

Invalid can also often mean that the test engineer doesn't understand the application under test properly in which case the test engineer should:

- Seek to clarify their understanding of the application
- Reflect on how the lack of understanding occurred
- Identify what further training or review might be needed now and in the future

The other alternative is that it is in fact VALID and the developer is stating a you're-not-supposed-to-do-that response, in which case the test engineer should hold firm that as "it" can be done the development team should clarify the development requirements and application objectives. This INVALID bug may then become valid or deferred as an ENHANCEment.

2.3 Wontfix

With this status the bug is accepted but for some reason the developer has decided that a fix won't be delivered. This may be because of considerations such as:

- The Bug is of low severity or impact and doesn't warrant the developer time to fix it, possible in consideration of higher severity issues that need to be resolved in the project timescales.
- The fix may be complex and carry a high risk of impacting other areas of the code base, there for it may be left as is and flagged as a Known Issue.
- A subsequent release of software will essentially resolve the issue. The current record would ideally be set to DEFERRED to ensure it is tested for in the next release.

In all cases they require discussion with development team and other key stakeholders such as the Product Manager due to the fact the bug remains within the product being released.

2.4 Duplicate

While they happen, the test engineer should be mindful of checking the bug management tool before submission of new bugs to minimise duplication. Where it is a 100% duplicate the planning of test execution may need review to avoid wasting time and resource. There's an additional benefit. Where the issues is not a full duplicate but is a similar issue the records can often be linked in the management tool.

This will help guide the developer to identify a possible single root cause for the linked issues. It's common to find a root cause manifest itself as a bug or collection of similar bugs in many locations across the application. e.g. a function sat in a library called by many components, an integration issue that is permeating through multiple branches of linked components.

Also, duplicate or similar issues alert the test engineer to 'error sensitive' coding and a bit of inductive analysis can be undertaken to identify further error sensitive lines of code to try and locate additional duplicate or similar bugs. E.g. ask the developer where else in the product code this particular or very similar way of coding has been used. That'll be a great service to the development effort and improvement of product quality. Consider feeding this back into developer coding standard guidelines to eliminate the issue in future products.

In addition, the more people are working on similar functionality the more the chance of having duplicates. For mitigation, a recommendation is sharing information about newly found defects among all team members involved in testing a specific part of system, as well as having one person responsible for "confirming" defect reports. In the latter case, that person shall be familiar with the functionality in which they provide the judgement.

Also if previous defect it closed and when testing this DUPLICATE defect, you find the same problem then you need to mark it as fail.

2.5 Enhance

With this status there can typically be two situations. One is where the bug covers a known issue, involving for example the use of newer technology in one part of the application and not another. Resolving the bug therefor results in enhancing the functionality of the product, albeit in line with other areas already enhanced.

A second possibility is that the fix is complex in nature and will require the first time use of newer technology or addition of new functionality. In this case the correct response from the development team is to set the status to ENHANCE. It’s common for bug moved to ENHANCE to also be further moved to DEFERRED.

2.6 Not Reproducible / Works for Me

These are the most mysterious kinds of defects. It would be a reasonable assumption that most Test Engineers have experienced these responses to bug reports. Where things that did not work yesterday work well today for some unknown reason. What the Test Engineer must do is make sure they have done everything reasonably possible to reproduce the Bug again. Doing so would require the ability to strictly reproduce the conditions of the previous test.

Conditions such as configuration of the test environment and assets, such as data, in the same way as the original test. Conditions such as system specifications including Hardware, OS and Environment used. The Test Engineer should persevere with a bug receiving this response, provided they have everything correct as per specifications and a valid Test Case.

This type of response is a good example of why the Test Engineer should ensure that the Bug report details are as accurate an unambiguous as possible. The bug report can often miss key information such as product version, feature or other important information that allows reproduction of the bug.

2.5 Deferred

With this status bugs are accepted but not fixed in the current development activity, being held for inclusion in a subsequent release. An approach to resolving the decision about deferral is for them to be agreed by a review meeting with participation of Development, Testing, Customer Support and other key stakeholders.

Due to the risk involved in extracting code with the bug it's common for the bug to be released and notified to customers as a known issue. This is an example of why the decision should be taken with the involvement of stakeholders such as Support or Information Security.