DEFECT MANAGEMENT PROCESS AND METRICS

Author: Manish Modi
Process: Testing
Version Number: 0.1
Publication Date: 16/11/2010
Contents

1 Introduction 3
1.1 What is a defect and why it is required to be managed 3
1.2 Defect Management Principles 3

2 Defect Management Process 4
2.1 Defect Management Group 6
2.2 Defect Handling 6
2.3 Categorisation of Test Defects (Priorities and Severities) 7
2.4 Resolution Codes 7
2.5 Defect Review Meetings 8
2.6 Test Defect Log 8

3 Defect Metrics for Effective Project Management 10
3.1 Defect Density 10
3.2 Defect Cause Distribution 11
3.3 Defect Submission (Detection) against Closure 11
3.4 Distribution according to Severity 12
3.5 Priority v/s State Defect Distribution 13
3.6 Percentage of Defects Rejected 13
3.7 Percentage of Defects Reopened 14

4 Conclusion 15

THIS DOCUMENT CONTAINS 16 PAGES INCLUDING TITLE PAGE
1 Introduction

This Whitepaper documents the process to be followed to manage individual defect raised during a test phase and describe various defect metrics like defect density, defect cause, defect priority v/s state, defect detection against their closure etc.

1.1 What is a defect and why it is required to be managed

A defect is a discrepancy between expected and actual results of the given system. For example, an incorrect implementation of specification and missing of the specific requirements(s) from the software

Software defects are expensive. Moreover, the cost of finding and correcting defects represents one of the most expensive software development activities. It will not be possible to eliminate defects but we can minimize their number and impact on our projects. To do this test team need to implement a defect management process that focuses on preventing defects, catching defects as early in the process as possible, and minimizing the impact of defects. A little investment in this process can yield significant returns.

1.2 Defect Management Principles

The defect management process is based on the following general principles:

- The primary goal is to prevent defects. Where this is not possible or practical, the goals are to both find the defect as quickly as possible and minimize the impact of the defect.

- Defect measurement should be integrated into the software development process and be used by the project team to improve the process. In other words, the project staff, by doing their job, should capture information on defects at the source. It should not be done after-the-fact by people unrelated to the project or system

- Defect information should be used to improve the process. This, in fact, is the primary reason for gathering defect information

- Most defects are caused by faulty or inconsistent processes. Thus to prevent defects, the process must be altered
2 Defect Management Process

A good defect management process is the one that doesn't just fulfil formal steps and procedures but strictly ensure that defects are being handled in a well-appropriate and organized manner from the time they are discovered till their resolution. A well planned progress will always have priorities in itself regarding the value of a defect, so normally there are four types of defects:

One that needs to be resolved immediately
High Priority Defects
Normal Priority Defects and
Low Priority Defects

For all test stages, defects will be logged by testers on defect tracking system against a project name unique to the application and release. This name will be stated within the Test Plan. An illustration of the defect template (otherwise known as a Defect Log) is shown in paragraph 2.6.

The Development Lead will then be responsible for assigning these to the relevant Developer for investigation. Once resolved, the Development Lead will assign back to the Test Team.

A Defect will pass through the following stages:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted</td>
<td>Defect created by Tester and Assigned</td>
</tr>
<tr>
<td>Opened</td>
<td>Triage determined as a valid defect</td>
</tr>
<tr>
<td>Assigned</td>
<td>Development Lead has assigned to a Developer</td>
</tr>
<tr>
<td>Resolved</td>
<td>Developer has fixed defect</td>
</tr>
<tr>
<td>Closed</td>
<td>No further action required – Closed by Tester</td>
</tr>
<tr>
<td>Reopened</td>
<td>When the defect is retested but not ok</td>
</tr>
<tr>
<td>Duplicate</td>
<td>Duplicate record (reference back to other defect)</td>
</tr>
<tr>
<td>Postponed</td>
<td>Defect put on hold</td>
</tr>
</tbody>
</table>
The life-cycle described above is just an example as to how a defect can be managed across its life cycle. It is purely a customisation based on organisation needs, several new states can be introduced or some of those described above may be removed from the defect life cycle.

If an unexpected occurrence is observed during test execution that is not scripted, a new script should be produced and the defect logged against that script. Testers will be responsible for ensuring that any relevant screenshots are attached to the defect record.

A defect cannot be closed without agreement between developers, testers and in some cases business analysts (i.e. where clarification of business requirements interpretation is required). Defect raised by the Client during UAT will be assigned after initial classification, usually to the Project Lead Tester.

All defects that are fixed during UAT must be successfully retested in System Test environment prior to release back into UAT. All of the defects that are identified must be captured in defect tracking system. Those that are deemed to be defects must be accurately described and categorised, fully analysed for impact and the status updated when it changes.

The defect management process will be controlled by a project specific Defect Management Group (DMG), as referenced in the following section.
2.1 Defect Management Group

2.1.1 Overview of the Defect Management Group
Defect Management is under the control of the Defect Management Group. The tasks and activities of this group are described in this section.

2.1.2 Composition of the Defect Management Group
The group members are appointed and managed by the Project Manager with the exception of the chairman who is appointed by the Client. Experts may be co-opted into the group to provide knowledge for specific test types (e.g.) business experts for UAT testing.

2.1.3 Client Representatives on the Defect Management Group
Resource planning has to be integrated into the general set up of the test organisation; but in practice, from the Client perspective, for SIT and UAT it will involve only the Client Test Manager or deputy on a day-to-day basis with expert business users (SMEs) consulted on specific points within their respective areas of competence. For OAT and Live Verification, it will also involve the Client Release Manager.

2.2 Defect Handling

The Defect Management Group (DMG) is used to categorise, classify and prioritise defects. When a defect is raised during test execution:-

- the DMG investigates to determine the cause
- If the defect is believed to be caused by an error in code or incorrect configuration of the infrastructure, the defect is deemed valid.

For valid defect:-

- the status is set to "open"
- the severity and priority codes are re-evaluated
- If the defect has been caused by an error in the code, it is assigned to the Development Lead
- If the defect has been caused by incorrect configuration or problem in the infrastructure, it is assigned to the Environments Team or deployment team if it is application related configuration
- Details of Priority 1 defects must be communicated to the Client and Project Manager or his deputy on assignment by telephone or e-mail
- Priority 2 defects should be communicated by email only
- a priority code assigned by the tester based on the impact of the defect on the testing process
- a severity code which is determined by the impact that the defect would have on the delivered system if it was not fixed
- both priority and severity can be re-assigned by the DMG
- the defect is then communicated to the appropriate team

If the defect has been deemed invalid:-

- the defect is marked with a status of closed with a resolution code of “raised in error”
- the circumstances that led to the erroneous defect report will be investigated.

If the cause is found to be an error in the testing process:-

- the incorrect script will be changed
- the defect will not be closed by the DMG until the changes are complete.
The test manager will be advised by the DMG on re-testing of fixed defects in consultation with the release manager.

If the test script or expected result proves to be incorrect, appropriate changes will be made and this may involve change control if the test script is misconceived or incorrectly scripted and the test set, to which the script relates, is run again.

### 2.3 Categorisation of Test Defects (Priorities and Severities)

Priority level informs the impact on the testing process and therefore determines how urgent it is to fix a defect that testing identifies.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prevents any meaningful testing being carried out</td>
</tr>
<tr>
<td>2</td>
<td>Stops a significant area of testing from progressing to completion</td>
</tr>
<tr>
<td>3</td>
<td>Stops a specific test being completed</td>
</tr>
<tr>
<td>4</td>
<td>Defect which does not stop a test being run to completion</td>
</tr>
<tr>
<td>5</td>
<td>Query resulting from an unexpected occurrence that does not impact on test</td>
</tr>
</tbody>
</table>

Severity level is informed by the impact of the defect on the system if it is not fixed.

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Critical – System cannot go live</td>
</tr>
<tr>
<td>2</td>
<td>High – The defect is significant with part of the system not working as specified</td>
</tr>
<tr>
<td>3</td>
<td>Moderate – the system does not meet stated requirements and impacts on the use of a significant part of the system</td>
</tr>
<tr>
<td>4</td>
<td>Low – does not meet stated requirements but there is an acceptable work-around available</td>
</tr>
<tr>
<td>5</td>
<td>Minor - does not meet stated requirements but has no adverse impact on the use of the system</td>
</tr>
</tbody>
</table>

### 2.4 Resolution Codes

Every defect must be closed down in defect tracking system with an associated Resolution Code from the list of valid values:-

- Change Request/Next Release
- Code Error
- Data Error
- Environment Build Problem
- Error already in production
- Raised in Error
- Tolerate

In the case of Live Verification defects raised by the client (business user), these will be raised in IT Service Management (ITSM) tool and the resolution codes will be in accordance with other Live defects. ITSM is configured with the following codes for ‘Cause’ in the Resolution Details:

- Capacity
- Change
2.5 **Defect Review Meetings**

During Test execution, regular (usually daily) meetings will be held between the Project Manager, Lead Tester, the Client’s Project Manager and Business Representative (DMG Group). Other parties, e.g. System Architects or Business Analyst may also be consulted on specific issues.

The main objectives of the meeting will be to:

- Review all new application and environment defects raised in order to agree that they are correctly categorised and prioritised.
- Monitor current status of all un-resolved defects
- Discuss defects which are obstructing tests
- Discuss requirements for any unscheduled code deployments or data refreshes to the test environment from live

In the event of any disagreement between the parties regarding the categorisation of defects, the matter shall be resolved in accordance with the provisions of the Dispute Resolution Procedure.

2.6 **Test Defect Log**

The following screenshot from **Defect Tracking System** (Rational Clear Quest) illustrates the Test Defect Log used. Every defect raised during testing will have an associated defect record logged and will follow the defect management process detailed earlier in this document.
3  Defect Metrics for Effective Project Management

Defect Metrics drive information from defect data (raised in defect tracking system) with a view to help in decision making. We can use the following defect metrics for effective project management

3.1  Defect Density

Defect Density is the number of confirmed defects detected in software/component during a defined period of development/operation divided by the size of the software/component

The 'defects' are:

- Confirmed and agreed upon (not just reported).
- Dropped defects are not counted.

The 'period' may be for one of the following:

- for a duration (say, the first month, the quarter, or the year).
- for each phase of the software life cycle (say testing, it may also be a specific testing stage like System Integration or User Acceptance Testing).
- for the whole of the software life cycle (Requirement and Design Reviews, Testing etc.)

The 'size' is measured in one of the following:

- Function Points (FP)
- Source Lines of Code

**Defect Density Formula**

\[
\text{Defect Density} = \frac{\text{Number of Defects (at given time)}}{\text{Software Size (FP/LOC)}}
\]

Let say we have found 3 defects during System Integration Testing (SIT) phase in 1000 lines of code but organisation data says 5 defects should be found from 1000 lines of code, it indicates we require further testing in SIT phase. On the other hand, if we find 8 defects in 1000 lines of code, it will be interpreted as system is quite unstable and it is required immediate attention.

We can use Defect Density for the following

- For comparing the relative number of defects in various software components/subsystem so that high-risk components can be identified and resources focused towards them
- For comparing software/products/releases so that 'quality' of each software/product/release can be quantified and resources focused towards those with low quality
3.2 Defect Cause Distribution

This metrics provide information about the defect root cause i.e. the reason by which defect exists in the software. Finding the root cause of the defects help in identifying more defects and sometimes help in even preventing the defects.

![Defect Distribution across Root Cause](image)

The above chart shows that maximum defects were caused due to Environment and User Role Permission. Accordingly Project Management team will take preventive action to reduce these defects.

3.3 Defect Submission (Detection) against Closure

This metrics provides information about detection of defects against their closure (cumulative). This comparison provides us vital information about software quality at any given point, prediction about remaining defects and time to fix remaining defects.
Defect Submission against Closure

The chart above shows that the test team identified most of the defects at early stage, detection of the defects at later stage was gradually decreased. Development team managed to close out the gap between submitted and closed defects towards the end.

If the gap between Submitted and Closed defects is not closed out then it indicates defect fixing activity is not effective and there are more defects introduced in the system.

If the submitted defects increase gradually towards the end of the project, it means that the test team identified most of the defects at the end of the life cycle.

3.4 Distribution according to Severity

This metrics provides a number of defects against each Severity. If there are a lot of critical severity defects, it will be a question on software quality and project management can take drastic steps to improve the software quality.
3.5 Priority v/s State Defect Distribution

This metrics provides information about defect count with state against each priority. If there a lot of P1 and P2 defects with Submitted/Assigned/Open state then Project Management will ask development team to work first on fixing these defects. By this metrics, project management team also find out closed defects count for any given priority.

3.6 Percentage of Defects Rejected

This metrics indicates the level of business understating among the testers. If the percentage of defect rejection is large then we can come on conclusion that tester doesn't have sufficient business knowledge.
3.7 Percentage of Defects Reopened

This metrics indicates the number of improper defect resolutions. There will be more unplanned efforts require re-fixing and testing these defects. It is an indicator on development process, it means how mature organisation development processes are.

At last not the least, I can say defect metrics are an **indicator of quality of a software/product under test and test team’s effectiveness**. These sets of metrics are equally important to track and base important test management decisions on, these metrics helping a test manager decide “whether the test team is ready to sign off on the software under test”. Also, as a best practice, I would recommend using these metrics on an ongoing basis (say once in 15 days or so) to keep track of how the software/product quality is shaping up. This will help the project or test management proactively recommend any corrective actions to the development team as opposed to reactively accepting poor quality later in the game.
4 Conclusion

We can use the following lessons which we learnt by experience to prevent or early detection of defects:

- System Integration Testing should be scheduled so that the core functionality are initially tested.
- Client responsibilities should be clearly communicated in the beginning of the project.
- Design documents should contain all the necessary validations to avoid validation error.
- Checklist should be used to avoid GUI errors.
- Rigorous unit testing to be done to avoid logical errors.
- Basic level review and testing should be done at developer’s level before handing over the code to testers.
- Test Cases should be formed with the valid test data.
Please send me your Feedback and Queries on

manish.modi@capgemini.com