International Journal of Academic Research and Development
ISSN: 2455-4197
Impact Factor: RJIF 5.22
www.academicsjournal.com
Volume 2; Issue 6; November 2017; Page No. 921-924

Bug tracking software

Nandha Kumar M
Assistant Professor, Department of Computer Science, AJK College of Arts & Science, Coimbatore, Tamil Nadu, India

Abstract
While there is no harm in using an excel sheet to record/track and emails to report/alert/communicate – as the magnitude of the projects, the number of test cycles, the count of the people involved grows – it becomes absolutely important that we need a much stronger mechanism that will make the management of these issues simpler and consistent so we can concentrate harder on actually finding more issues in the AUT than managing the ones already found. o enable the same, the QA market has seen the emergence of various bug tracking systems or defect management tools over the years. As is the general rule, all the tools that belong to a certain 'genre' consist of certain common/similar features that we can bank on.

Keywords: tracking, software

Introduction
Defect cycle or defect life cycle is ride of a defect from discovering defect to closure of defect. Defects management in defect cycle is important to ensure the software quality. Preventing, identifying, rectifying defect is important to improve the quality. Defect is managed and tracked easily throughout the defect cycle with the use of defect tracking tools like JIRA, Mantis, Team Service, Bugzilla, Redmine etc. In a project identifying defects in early stage and fixing will take less cost when compare with identifying and fixing defects in later stage of the project. A tester need to identify valid defect and raise the defect with all the detail and make sure the defect gets closed. We also need to make sure the closed defect is verified and in the close stage in future releases also. If it gets reopen that should be resolved, verified and closed again. Defect have several stage from identification to closure. Managing all the defects on particular software project is important to complete the project successfully.

Stages of defect in defect life cycle
Identifying defect
To identify defect, the tester should know exactly what the expected behavior of the system is. While doing the test execution tester will compare with the expected behavior and if it vary, there is a defect discovered. Tester should think is non-functional areas such as usability, user friendly also and if there is any problem in that that also need to be raised as a defect or communicate as an improvement. Always tester should make sure the exact expected result is mentioned along with the defect when report the defect. Tester should be concern and raise the valid defects. Defects that hold the state New, Assigned and Reopened are falls under this stage. After this stage defect shall be resolved.

Resolving Defect
Tester and developer should rectify the defect without the negative impact to other areas. While resolving the defect, the expected behavior should work correctly and defect should not be there anymore. Defects that hold the state fixed, duplicated, cannot reproduce, cannot resolve and not a defect are under this stage. After this stage the defect is ready to retest.

Retesting Defect
After developer rectifies the defect, tester will retest the particular defect. Tester will test whether the defect is fixed and the expected behavior of the system is applied. Defects that hold the state fixed, duplicated, cannot reproduce, cannot resolve and not a defect will be verified in this stage. Once retested the defect transfer either to reopen or closed state. Defect's journey is end when the defect is closed. But the defect should be verified in later releases also to check the expected behavior is working and defect is no more occurring.

States in defect life cycle
New
When a defect is encountered in the software project, the tester is supposed to raise it. At the very first time when the defect is raised, the stage of the defect is 'New'. Tester should always analyze and make sure that it is a valid defect. Tester should know what is the expected result whenever encounter a defect.

Assigned
Once the defect is raised it will be assigned to the developer to resolve the defect. Tester should know to which developer the particular defect should be assigned. Once assigned the defect is transfer to the developer hand.

Fixed
Developer will analyze, check and fix the defect. While fixing the defect developer should make sure there is no other negative influence to other areas in the software.

Deferred
There are some instance the defect is valid but it will be fixed in the future releases, then the defect transfer to deferred state. As a good practice when deferred the defect, the decision needs to be taken after discussing with the team members.

Duplicate
There are some cases the defect is valid but it's been already reported either by another tester or in another form. At this time developer will mark the defect as duplicate defect.

Not a defect
If the defect is not a valid defect then the developer will state the defect as 'Not a defect'.

Cannot Reproduce
There are some instance the developer can't reproduce the defect, then the developer will mark as cannot reproduce. It's better to check with the tester and try to re-create the defect. Some defects are data specific or environment specific. At this time it's good to check with tester and find the root cause and resolve the defect.

Cannot Resolve
There can be a defect which cannot be rectified due to any limitation like technical limitations. At this time the defect goes to the state cannot resolve. When we cannot resolve a defect, we need to communicate that to the team and if need this needs to be communicated to the client or end users as well.

Reopen
After the state of fixed, duplicate, deferred, cannot reproduce, and cannot resolve, the defect transferred to tester's hand again for verification. Tester should retest whether the defect is fixed or not. If the defect is not resolved then the defect will be reopened. While reopen the defect again the defect goes to developer's hand.

Closed
While tester retest the defect if the defect is fixed then the defect goes to 'Closed' state. Tester should verify the closed defect in future and make sure the defect is not happening any more in the software. If the closed defect is encountered again in the software, it will be reopen and travel from assign state again.

Defect life cycle
Below diagram explain the flow of the defect throughout the defect life cycle.

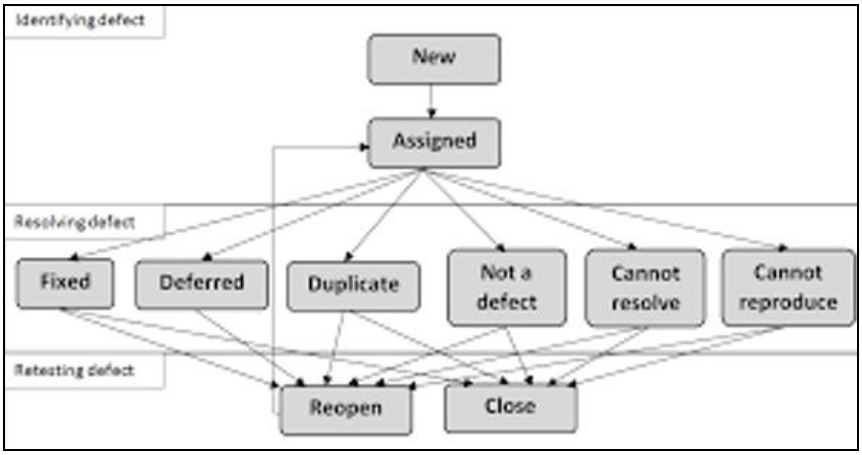

Fig 1

Defect Management
First of all it is important to prevent defects as possible in software. Developer should keep this in mind while developing. Testers should discover the defect as quick as possible. Cost and the effort is less to fix the defects identified at the early stage in the software development life cycle. Once the defect is discovered it will be recorded and tracked. Defect is recorded in detail with all the required details. Defect tracking is important to close the defects and reach the appropriate quality. Then the discovered defect should be rectified properly and maintain that it will not occur again in the software. Defect is analyzed and root cause will be find, to improve the development and software quality. Defects and defect reports shall be analyzed and help to improve the software quality. It also helpful to improve the individual team members. They could improve their skills while analyzing the defects.

3 Intangible Benefits of Using a Bug Tracking System
Firstly, Why to Use a Defect Tracking Tool?
In the absence of a Bug Tracking Tool, teams use spreadsheets to report, track and transport their bugs. While this might be a good temporary solution for small sized teams and projects, this is not a sustainable method.

Here is why.
Spreadsheets/Excel sheets pose a ton of challenges when you use them as your primary method of defect tracking and management.
To name a few of them listed below:

1. Too many bulky emails: Does this ring a bell? Excel sheets with screenshot attachments are sometimes over a few MB.
   I often had these spreadsheet attached emails sitting in my outbox waiting to be sent or receiving a mailbox full alert

as soon as I got one.

2. Lack of real-time visibility into bug discovery and progress/status: We don't hear of an issue as soon as it is found. We also don't know if an issue has been retested or returned, etc. real time.
   Since there is no automatic alerting system, defects do not call for any attention to them unless someone is deliberately looking.
3. Work assignment issues: We don't know who has what issue and what they are doing. If it has been picked up for resolution, what priority is set, etc. is never as easily visible as you would like it to be.
   You might have to call or email or send an IM to find out what is happening.
4. Lack of a central repository: Too many folders, release-wise, module-wise or something-else-wise.
   If you want to get back to a defect that was reported in the previous release or maybe a few releases behind, which was commented on by the developer in a certain way- you are simply playing a guessing game as to where the defect might be.
   Even if you did find it, you might not have all the comments on it, all the history of it, etc.
5. Manual gathering and consolidation of defect statistics for insights into Quality of the product.

Imagine collecting raw defect data from each team member, entering into an excel template, organizing it to show a pattern or trend, and finally plotting a chart or graph. This process is time and labor intensive. And also, rigid.

Say if your team wants to view a new kind of report, you are looking at the additional effort and creating new templates, etc. So you are really limited your choices of what defect trends you can and will see.

Teams will no longer be inclined to invest time in monitoring and measuring and this means lack of visibility and confidence about the quality of the product.

Some problems could be solved with the use of a shared document on a remote/network drive, but not all. So, most test teams use a defect tracking tool to handle this process effectively.

Defect management/bug tracking tools offer a single point of truth for all your defects, provide real-time updates, aid collaboration with the team members, trace the defects back to the requirements and generate real-time reports.

Everyone knows about this, what's new?

Here are some great ways you can make your bug tracking tool multi-task.

3 intangible benefits of using Bug Tracking Systems

I guarantee you that the defects in your report will be superior, valid, and easier to understand and will have a higher 'picked-for-resolution' rate.

2) Understand defect reporting standards

Now, every company, every project, every team and every individual are different. So, even though a few common guidelines on how to write defect reports exist, nothing prepares you like your own in-house research does.

How do you do that?

Check your defect tracking tool for the following:

- What defect reports got returned as "Not enough information"?
- What defects were outright rejected by developers as 'Not a defect' or 'works as intended'. And, why?
- What enhancement suggestions were considered?
- What defects are still open?
- Did reports with screenshots have a higher rate of being fixed?
- For a defect, if the developers changed the severity, check out why? This might let you know what is 'serious' for the team and what is not.

3) Prevent duplicates and invalid suggestions

Once you know your application, your team's work style, your development team better you are automatically a better tester.

This way you will know what is already reported or what has already been suggested and rejected.

You can now focus your energies on uncovering new bugs, exploring the application deeper and tailor your reports in a way that you get through to your development team better.

15 Most Popular Bug Tracking Software

Bug Tracking Tools

Web Based Bug Tracking

Table 1

Product	Vendor	Comments
Bug/Defect Tracking Expert	Applied Innovation Management	Web-based bug tracking software
Bugzilla	Bugzilla.org	Highly configurable Open source defect tracking system developed originally for the Mozilla project
Bugzero	WEBsina	Web-based, easy-to-install, cross-platform defect tracking system
DefectTracker	Pragmatic Software	Subscription-based bug/problem tracking solution
FogBUGZ	Fog Creek S/W	Web-based defect tracking. Free eval, 90 day money back guarantee.
JIRA	atlassian	J2EE-based, issue tracking and project management application. Extensible via Java API.
Mantis		Lightweight and simple bugtracking system. Easily modifiable, customizable, and upgradeable. Open Source.
	Active-X.COM	BugHost is a feature-complete hosted defect tracking system ideal for small- to medium-sized companies who want a secure, Web-based issue and bug management system. There is no software to install and can be accessed from any Internet connection. Designed from the ground up, the system is easy to use, extremely powerful, and customizable to meet your needs.

MyBugReport	Bug Tracker	Online testing tool that allows the different participants working on the development of a software or multimedia application to detect new bugs, to ensure their follow-up, to give them a priority and to assign them within the team.
PR Tracker	Softwise Company	Records problem reports in a network and web-based database that supports access by multiple users. Features include classification, assignment, sorting, searching, reporting, access control, & more.
Task Complete	Smart Design Te	TaskComplete enables a team to organize and track software defects using with integrated calendar, discussion, and document management capabilities. Can easily be customized to meet the needs of any software development team.
Agility	AgileEdge	Issue and Bug Tracking Software. Agility features a easy to use web-based interface. Key features include fully customizable field lists, workflow engine, and email notifications.
BUGtrack	SkyeyTech, Inc.	Web based defect tracking system
Fast BugTrack	AlceaTech	Fast BugTrack is a market leader in bug tracking solutions. Completely Web-enabled, quick to install, and easy to use, Fast BugTrack offers you the ability to track bugs, coordinate projects, and effortlessly manage the change process within your organization.

Conclusion

Defect Management system, when used right – as a tester, you understand your ecosystem better and as a team, it will improve the overall efficiency. Therefore, if you are still using the primitive spreadsheet method for bug tracking, it's time to change. The choices for bug tracking tools are many. If using a test management tool you are going to have access to defect tracking as well.

Some companies create in-house bug tracking tools. They are similar to the commercial ones available. So, they do the job just fine. Commercial, yet affordable tools. E.g. JIRA or FogBugz

Finally, if all your team needs is a tool for defect tracking and if the entire testing is still maintained manually, your best option is to go with an open source defect management/bug tracking system.

References

1. Http://www.softwaretestinghelp.com/practical-implementation-of-manual-testing/
2. A comparative study of automated software. Testing tools. Nazia Islam. Nazia Islam. isna1301@stcloudstate.edu
3. Issn-2319-8354(e). 173 | p a g e. Comparative study of software. Testing tools on the basis of software. Testing methodologies.
4. A study on Various software Automation Testing Tools Neha bhateja, Department of computer science, Amity university Gurgaon, Haryana, India
5. https://www.guru99.com/testing-tools.html#9
6. International Journal of Information and Computation Technology. 2013; 3(7)711-716. ISSN 0974-2239 © International Research Publications