ISSN (Online): 2230-7893
www.IJCEM.org

Reducing Duplicate Defects

Sandru Veerabhadraiah[1], Venkat Ramesh Atigadda[2], Rama Murari[3]

[1,2,3]Hi-Tech ISU – Assurance CoE, Tata Consultancy Services Limited

SEZ Unit, Deccan Park, Madhapur, Hyderabad, India.

Abstract

Testers or users submit bug reports to identify various issues with applications. Sometimes two or more bug reports correspond to the same defect. To address this issue on duplicate defects we came up with multiple approaches which help in reducing duplicate defect logging

Key Words: Duplicate Defects

1. Introduction

Software runs our lives today. We use hundreds of applications without noticing them and due to the development of new features and periodic enhancements, applications are becoming large and critical; there is a need for an effective testing so that more number of defects were logged to ensure quality.

According to analyst Nelson Hall, the global testing services market is $8.4bn for the current fiscal year, it predicts an average 9% growth every year over the next five years and Indian-offshore based delivered testing services are grown by 8% while onshore-delivered testing services are to decline by 5% [7]. Cost savings are a major driver and also the need to do testing in a more professional manner, MOZILLA had received more than 420,000 defects out of which 30% of defects are duplicates and the cost invested on each duplicate defect is $75[1, 5].

As per the survey conducted by Casper Jones & Associates [1], the below table provides the complete view of the percentage of duplicate defects befall in the industry under each category. In this article, we present some of the major causes of duplicate defects and propose different approaches/suggestions to reduce logging of duplicate defects.

Figure1 illustrates some of the causes of Duplicate Defects

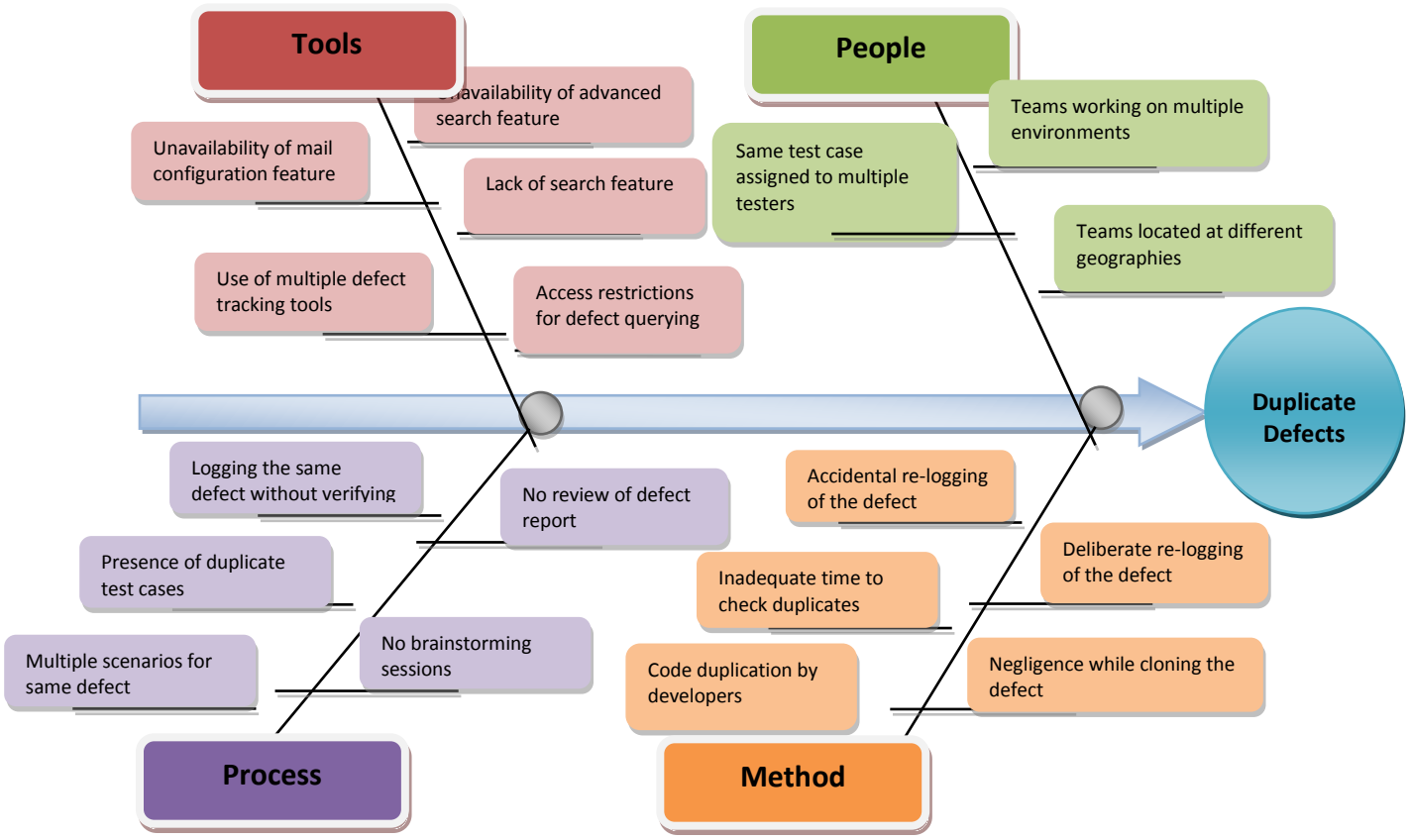

Figure 1: Causes of Duplicate Defects

S.No	Defect Category	Percentage of Duplicate Defects
1	Code Defects	35
2	Structural Defects	20
3	Data Defects	20
4	Requirement Defects	6
5	Design Defects	6
6	Bad Fix Defects	6
7	Requirements creep defects	3
8	Test Case Defects	2
9	Document Defects	2
Total Defects		100

Table1: Percentage of duplicate defects in each category

2. Typical Causes of Duplicate Defects

Duplicate defect causes effort leakage for all the stake holders as the defects need to be discussed during triages and also some effort spent on raising the defects, analysis etc.

Node	Cause	Details
Tools	Unavailability of mail configuration feature	Mail configuration feature helps the team in identifying the defects logged by other team members, so that duplication can be avoided
	Unavailability of advanced search feature	Advanced search feature like searching for duplicates based on comments can be used to avoid logging of duplicate defects
	Lack of search feature	Lack of search feature in the defect tracking tool will consume more effort for the testing team to check for duplicate defects manually
	Access restrictions for defect querying	Testers not able to use the search feature due to access restrictions.
	Use of multiple defect tracking tools	Many of the projects have multiple clients, and there are chances that each client use different tools for logging defects, which makes it impossible to find duplicate defects
People	Same test case assigned to multiple testers	Re-assigning the same test cases to multiple team members, causes defect duplication
	Teams working on multiple environments	Teams executing the same test case in multiple environments, may end up in logging the same defect
	Teams located at different geographies	Testers testing the same application from the different geographic locations may end up in logging the similar defect which was already logged by the tester from some other location
Process	Logging the same defect without verifying	Logging the same defect without verifying whether defect is already logged, will result in duplication
	Multiple scenarios for same defect	Testers tend to log multiple defects for the same scenarios with slight variation in steps
	Presence of duplicate test cases	Having duplicate test cases in the test suite will lead to duplication of defect when the same test case is executed within the team
	No review of defect report	No internal/External review of the defect report by the SME's may cause defect duplication
	No brainstorming sessions	No internal discussions on the defects raised by the team will result in duplication of defects
Method	Accidental re-logging of the defect	Testers sometimes accidentally click on submit button multiple times which result in duplicate defects
	Deliberate re-logging of the defect	Testers tend to re-submit the previously logged defect again, if the previous defect is not addressed by the developer for a longer period
	Code duplication by developers	Copy/Paste of existing code by the developers will result in duplicate defects
	Negligence while cloning the defect	Testers try to clone the existing defect, may re-submit the same defect without making the changes
	Inadequate time to check duplicates	Due to insufficient time, testers are not able to check for the defects which are already reported by the team

Table 1: Detailed description of each cause

3. Approaches to Reduce Logging of Duplicate Defects:

3.1 Reducing of Duplicate Defects through Mails

Due to the continuous usage of Defect Tracking tools by different clients across the globe, the tool vendors are constantly enhancing the tool features to make it more user friendly. Mail configuration is one important feature which is widely used by the clients. This feature help the team to configure different mail servers in order to receive mail notifications if any new defect has been added or any changes are made to the status of the existing defect in the defect management tool. We can further filter the mail receiving options so as to receive mails only for a particular module/product in the application, so that Testers/Team can receive mails for any changes to the defects which were logged under that particular module.

Figure 2 gives the pictorial view of the approach and helps the team to understand and visualize the advantages of receiving the mail notifications for the defects. When the defect is found, the tester can easily check the mails received from the other team members before checking for the duplicates in the defect repository. If the similar defect is not found either in the mails (based on the subject line/summary of the defect) or in the repository then the tester will go ahead and log it as a new defect. This approach will definitely avoid re-work and save time for the reviewer or developer in analyzing the defect.

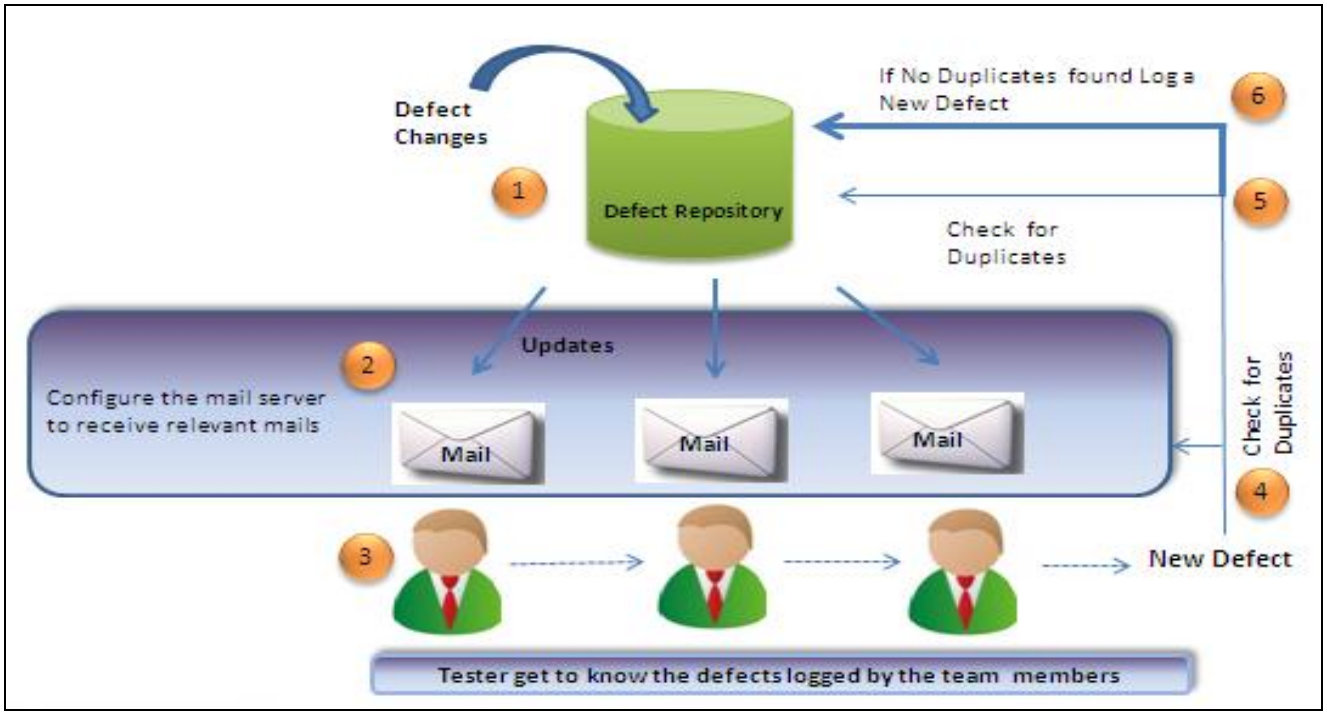

Figure 2: Reducing Duplicate Defects through mails

Table 3 shows the sample email notification for the defect. Subject line or Summary of the defect will help the testers to identify the similar defects which were logged by the other team members. With this approach the team can avoid logging of duplicate defects.

	<<Summary of the Defect>>
	<<Name of the Project>>
	<<Name of the Product>>
	<<Name of the Component>>
	<<Functional Area of the Identified Defect>>
	<<Version of the Application>>
	<<Current status of the defect>>
	<<Defect Link which will invoke the defect in the defect tracking tool>>

Table 3: Sample E-mail Notification of the Defect

3.2 Reducing Duplicate Defect Logging Using Internal Review

Team needs to extract all the open defects from the defect repository to the Internal Repository or spreadsheet and any new defect found by the team should be checked against the existing defects before updating the defect in the spreadsheet. The internal report is further reviewed by the reviewer on the daily basis before instructing the team to log the defects in defect repository. Additional review from the reviewer will make sure that no duplicates are reported. Reviewers are more skilled in detecting duplicates and they also know the system better. For the easy review, spreadsheet should have an attribute to determine the functional area /module in which the defect was found and use of filters in spreadsheet will narrow down the search for duplicates. **Figure 3** illustrates the sequential process of this approach

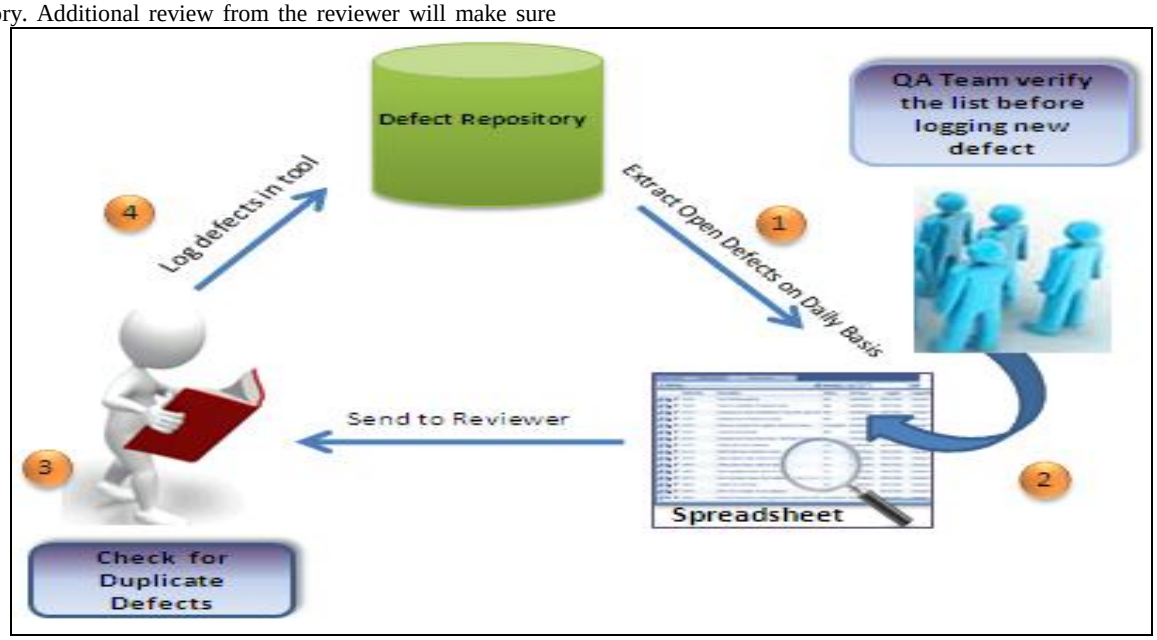

Figure 3: Reducing Duplicate Defects using Internal Review

Table 4 provides the sample view of the Defect Duplication checklist, will help the team to check for the similar defects on different functional areas before logging, so that duplication can be avoided.

S.No	Checklist Activities	Priority	Yes	No	Remarks
1	Check the defects which are logged for the same Requirement ID	High			
2	Check the defects which are logged under the same Test Case ID	High			
3	Check for the defects which has the same key words	High			
4	Check for the defects which are logged under the same Module	High			
5	Check the defects which are logged under the same Component	High			
6	Check the defects which are logged under the same Functional Area	High			
7	Check for the defects which are logged under the same Version	Medium			
8	Check for the defects which are logged under the same Release Environment	Medium			
9	Check for the defects which were assigned to same developer	Medium			
10	Check for the defects which are logged under the same Product	Medium			
11	Check for the defect which was logged under the same Operating System	Low			
12	Check for the defects which were logged for the same target cycle	Low			

Table 4: Sample view of checklist

3.3 Finding Duplicate Defects Using Natural Language and Execution Information Approach

Using Natural Language approach we can identify the duplicate defects based on key word search, before logging any defect we can choose the key words from the summary of the defect and search for the matching words from the repository. With this approach there is a high possibility of finding the duplicates [2].

Execution Information approach is more depended on the Root Cause analysis of the defect, which will help in finding the actual cause of the defect. The cause, if corrected should prevent recurrence of the defect. Based on the similar root causes, duplicate defects can be easily identified.

Both Natural Language and Execution Information approaches have the following advantages. First, natural language information acquired from the bug description will most likely represents the external buggy behavior observed by the tester, while the corresponding execution information likely records the internal abnormal behavior. Thus, we can leverage both these approaches in identifying duplicate defects.

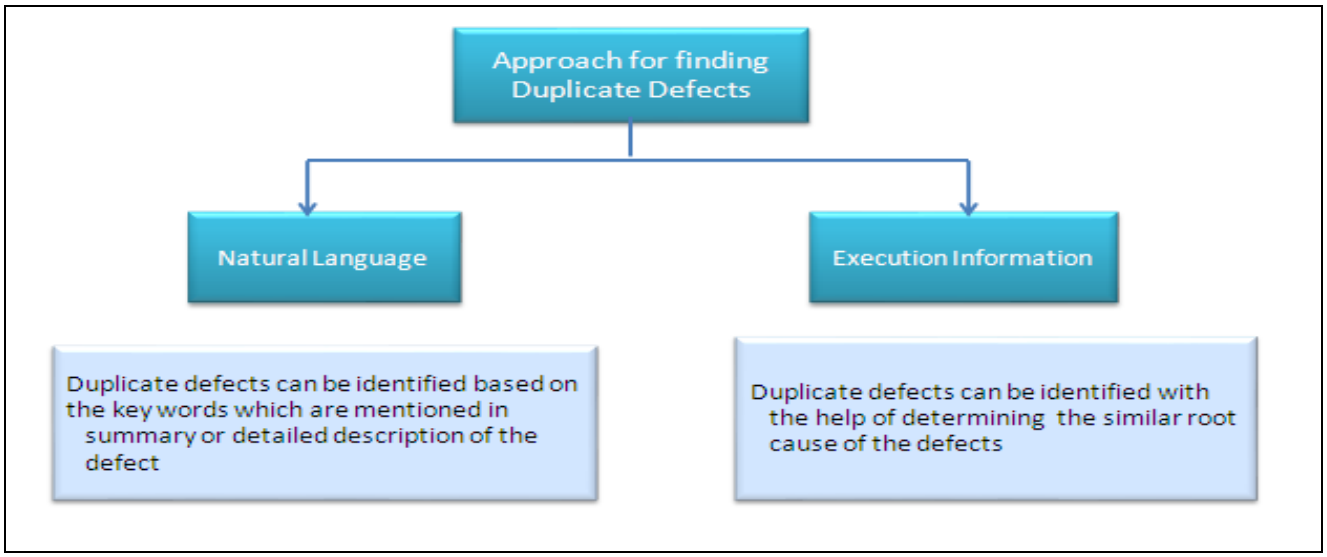

Figure 4: Natural Language and Execution Information approach

As the number of defects in a defect repository increases, there is a possibility that the number of potential duplicate defects also increases. However, it is challenging to manually detect all potential duplicates because of the large number of existing defects. JDF (Jazz Duplicate Finder) tool [6] finds potential duplicates for a given defect using natural language and execution information.

Jazz provides repository for all the work items which require coordinated team work such as defects. The defect attributes in Jazz are similar to other common defect tracking tools like Bugzilla etc.

While submitting the defect, Jazz tool automatically verifies the existing defects from the repository based on Natural Language and Execution Information and provides a list of similar defects. The below steps provide how the similar defects are identified

Step1	**Calculates the Natural Language based similarities (NL-S) between new defect and existing defect**
Step2	Calculates the Execution Information based similarities(E-S) between new and existing defect
Step3	Retrieve potential defects list using both NL-S and E-S

4. Usage of tools in finding the Duplicate Defects

All the software development projects require defect tracking systems and defect tracking systems play a vital role in the maintenance activities of software developers

4.1 Using Quality Center (QC)

Quality Center (QC) is considered as one of the best Test Management Tool in the industry; the tool has got variety of features and can be integrated with the other functional automation tools easily, one of the best features in Quality Center is 'Find Similar Defects'. When a new defect is logged, QC stores a list of key words which are mentioned in the summary or description of the defect and compares with the key words of the new defect.

Figure 5 shows the step by step process for finding similar defects which are logged in QC.

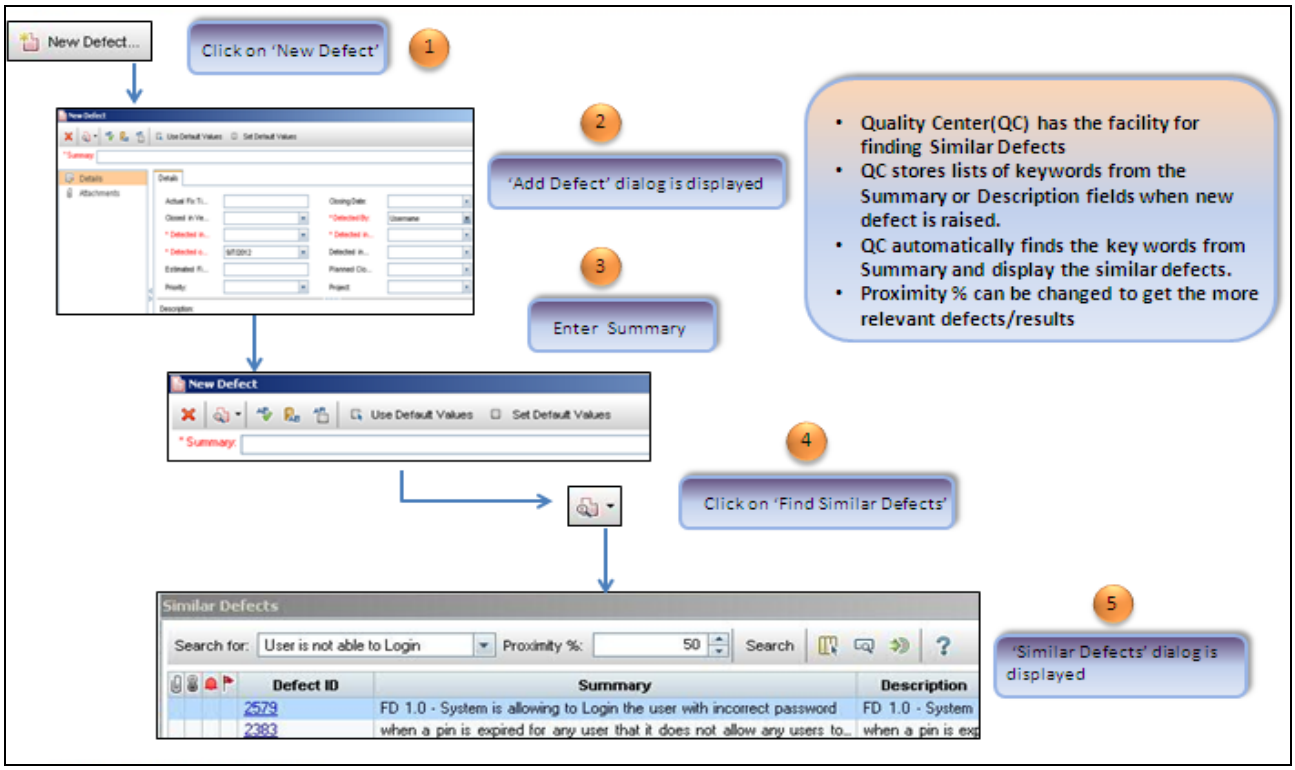

Figure 5: Detailed workflow for finding duplicate defects

Step#	Description
Step1	First, Click on Add defect button from the Defects tab
Step2	'Add Details' dialog is displayed
Step3	Enter the summary of the defect in the Summary filed
Step4	Click on "Find Similar Defects" option *<<"Similar Defects" window will be opened, which has the list of similar defects reported in the QC along with their Defect ID's >>*
Step5	Change the Proximity % option *<<Further filtration is done and only the closely match results are displayed>>*

Verify the results after filtration and check whether the defect is already reported to avoid re-logging of the similar defect.

4.2: Using Bugzilla

Bugzilla is the most popular bug tracking system available today. Bugzilla's popularity is due to its highly customizable interface, easy configuration, variety of features and a large community of very active users. Bugzilla has many important features, including the feature for finding the duplicate defects. Similar defects can be identified by entering the key words in edit box and the results can be further filtered based on Product, Component and Version. We can search for similar defects based on the comments entered by the tester or the developer and also based on the URL on which the defect was found. **Figure 6** gives the sample view of this feature.

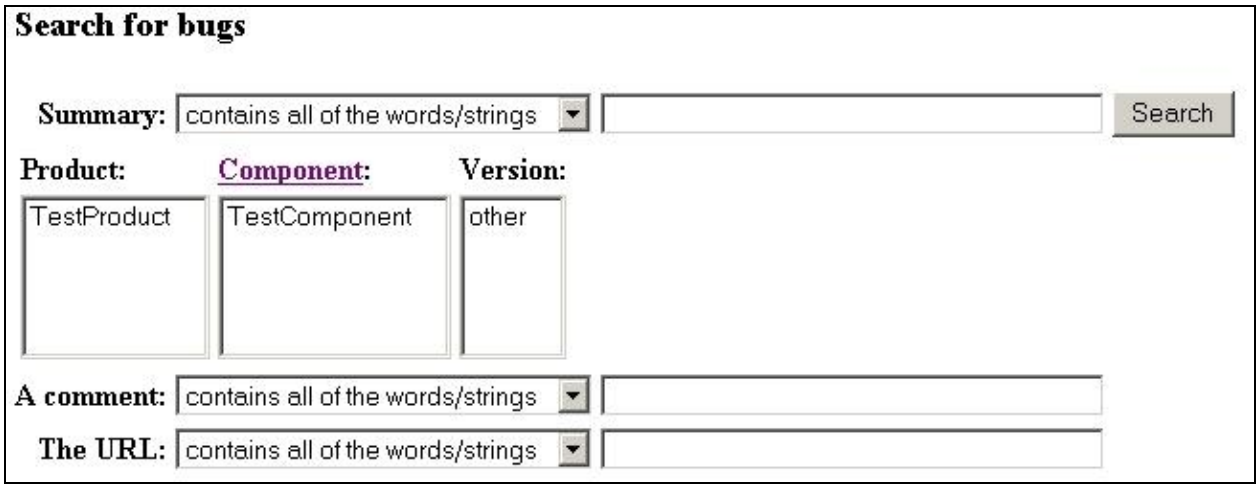

Figure 6: Finding Duplicate Defects in Bugzilla

5. Conclusion

In this paper we have presented different approaches and tools for reducing logging of duplicate defects. The central idea behind each approach is to save time and cost spent on unnecessary activities. Multiple approaches can be used in parallel to minimize the duplication of defects.

References:

[1]Software quality in 2011: a survey of the state of the art - *SQGNE*
[2]An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information - *Xiaoyin Wang, Lu Zhang, Tao Xie, John Anvik and Jiasu Sun*
[3] Duplicate Bug Reports Considered Harmful? - *Rahul Premraj and Thomas Zimmermann*
[4]Towards More Accurate Retrieval of Duplicate Bug Reports - *Chengnian Sun, David Loy, Siau-Cheng Khoo, Jing Jiang*
[5]Investing in Software Testing: The Cost of Software Quality
[6] JDF: Detecting Duplicate Bug Reports in Jazz - *Yoonki Song, Xiaoyin Wang, Tao Xie*
[7]Why software testing is increasingly being outsourced and what should businesses consider? - *Karl Flinders*

Author's biography

Sandru Veerabhadraiah has 16 years of experience in IT and 8 year's of experience in software testing. Currently he is leading the Assurance CoE of HiTech Industry Solution unit of Tata Consultancy Services. In this role, he is responsible for test offerings creation, proposal solutioning, competency enhancements and asset creation and deployment in the areas of software testing. He and his team also participates in testing projects technical reviews and recommends improvements in the areas of software testing. His area of expertise include test process consulting, test management, test automation and test methodologies in usability testing, globalization, accessibility testing.

Venkat Ramesh Atigadda is having 9.5 years of industry experience in Software Testing and worked for industry verticals like Energy and HealthCare. Currently he is a Solution Developer for Assurance CoE of HiTech Industry Solution Unit of Tata Consultancy Services. In his role, he is responsible for Asset Creation &Reviews, Test Strategy Consulting, Project Technical Reviews and analysis of latest testing Trends.

Rama Murari has 13 years of IT experience and is into Software Testing. From the past 7 years she has been in various roles of developer, business analyst, functional analyst, tester, project leader and program manager for large and renowned accounts. She is working as a Solution Developer in Assurance CoE of HiTech Industry Solution unit of Tata Consultancy Services. Her areas of expertise include Test Process Consulting, Test Management and White box testing.