International Journal of Electrical, Electronics and Computer Engineering **3**(1): 212-217(2014)
ISSN No. (Online): 2277-2626

Defect Tracking System

Sujata Solanke* and Prof. Prakash N. Kalavadekar**
**Research Scholar, Department of Computer Engineering, S.R.E.S, College of Engineering, Kopargaon, (MS), India*
***Assistant Professor, Department of Computer Engineering, S.R.E.S, College of Engineering, Kopargaon, (MS), India*

(Corresponding author: Sujata Solanke)
(Received 05 March, 2014 Accepted 7 June, 2014)

ABSTRACT: For the improvement of software quality now a days Defect Tracking System has been developed. There are various already existing methods like Redmine, Bugzilla etc. which doesn't meet the criteria of perfect defect tracker. This paper is aimed at developing an online defect tracking system useful for applications developed in an organization. The Defect Tracking System (DTS) is a web based solicitation that can be accessed throughout the organization. In this system can be used for sorting defects against an application/module, assigning defects to individuals and tracking the defects to resolution. This solicitation contains features like email notifications, user maintenance, user access control, report generators etc. This paper has been planned to be having the view of distributed architecture, with centralized storage of the database. The system for the storage of the data has been scheduled. Using the paradigms of MS-SQL Server and all the user interfaces has been designed using the ASP.Net technologies. The principles of security and data protecting mechanism have been given a big choice for proper procedure. The solicitation takes care of different modules and their related reports, which are created as per the applicable strategies and principles that are put forwarded by the administrative staff. This system will overcomes all problems of previously existing bug trackers.

I. INTRODUCTION

The Defect Tracking solicitation allows you carry out four important tasks finding bugs, changing bugs reporting, about bugs, application maintenance while using components. The role-based security mechanism implemented in the Defect Tracking System grants access to each of these features to the roles defined in the system. Each operator should be allocated to at least one role so they can log in and achieve one of these tasks. Defect management is critical to closing the loop between requirements, implementation and verification and validation. Archaic defect tracking management, implemented in a standalone fashion, can no longer address the difficulty and pace of change in modern software development. Defect management processes must be firmly interlinked with all of the other software development procedures. The defect management process contains the following elements:

A. Defect Discovery

Identification and reporting of prospective defects. The defect tracking software must be user friendly so that people will use it, but ensure that the minimum essential information is achieved. The information captured here should be enough to replicate the defect and allow development to determine origin and influence

B. Defect Analysis & Prioritization

The development team determines if the defect report relates to an actual defect, if the defect has already been reported, and what is the impact on development team. As an integrated development solution Integrity supports all other disciplines in the application lifecycle - not just defect management. Defect management software cannot achieve the seamless links among all activities and assets that are needed in today's fast-paced and difficult development environments. Integrity increases product quality and customer satisfaction by facilitating defect tracking across product variants. Relating these linkages with unprecedented process flexibility makes Integrity the best choice for and priority of the defect is. Prioritizing and scheduling of the defect resolution is often part of the overall change management process for the software development association.

C. Defect Resolution
Development team determines the source cause, implements the changes needed to fix the defect, and documents the details of the resolution in the defect management software, including ideas on how to verify the defect is fixed. In organizations using software product lines approaches, or other shared component methods, defect resolution may need to be coordinated across several branches of development.

D. Defect Verification
The build containing the resolution to the defect is identified, and testing of the build is performed to ensure the defect truly has been resolved, and that the resolution has not introduced side effects or regressions. Once all affected branches of development have been verified as resolved, the defect can be closed.

E. Defect Communication
This comprehends automatic generation of defect metrics for management reporting and process development purposes, as well as visibility into the presence and status of defects across all disciplines of the software addressing challenges such as:
(i) Discovery and Warning
(ii) Defect Determination across Multiple Lines of development
(iii) Automatic Defect Verification
Further paper is organized as follows: section 2, gives literature survey in which a short summary of existing solutions of the problem. Section 3, implementation details gives the brief idea of design procedures affected branches of development have been verified as resolved, the defect can be closed.

II. LITERATURE SURVEY

User complains the existing solutions from the point of their usability such as they couldn't recognize the developers or testers who created the defects. These already existing systems couldn't keep track of the already detected bugs. In general, the overviewed solutions give excellent results under the exact problem of interest, but they do not report the general problem of interest for this research.
It gives a brief for each of selected solutions, as per the following main points:
(a) The basic information of all solution,
(b) Exact details for each selected solution,
(c) Additional development trends of the approach, and
(d) A criticism of the solution, and finally
(e) Possible developments that could overcome the noticed drawbacks.
It determines with a classification of each expanded solution.
The classification criteria were selected to reflect the essence of the basic viewpoint of this research. It will summarize all significant parts of elaborated solutions.
Presentation of existing solutions and their disadvantages
This section is distributed in several subsections, one per each solution.

A. Bugzilla
Bugzilla is very popular, actively maintained and bug free tracking system, used and established together with Mozilla, giving it considerable authority. Bugzilla is based on Perl and once it is set up, it seems to make it user friendly. It's not highly customizable, but in an odd way, that may be one of its features: Bugzilla installations tend to look pretty much the same wherever they are found, which means many developers are already familiar to its interface and will feel that they are in familiar territory. Bugzilla has a system that will send you, another user, or a group that you specified. Bugzilla has very innovative reporting systems and you can create different types of charts with line graph, bar graph or pie chart.

B. Mantis
Mantis is a free web-based bug tracking application. It is in the PHP scripting language and works with MySQL, MS SQL, and PostgreSQL databases and a web server. Mantis can be installed on Windows, Linux, Mac OS and OS/2. Almost any web browser should be able to function as a client. It is released in the terms of the GNU General Public License (GPL). The main objection is its interface which doesn't meet modern standards. On the other hand, is easy to navigate, even for inexperienced users. There not exist some innovative features such as charts and reports. In short, the whole system is untidily done; there are plenty of bugs and very little functionality.

C. BugTracker.NET
BugTracker.NET is a free, open-source, web-based bug tracker or customer support issue tracker written using ASP.NET, C#, and Microsoft SQL Server Express. BugTracker.NET is easy to install and learn how to use. When you first install it, it is very simple to setup and you can start using it.
Later, you can change its configuration to handle your requirements. It has a very intuitive interface for creating lists of bugs. It has two useful features. First of them is a screen capture utility that enables you to capture the screen, add annotations and post it as bug in just a few clicks. The second feature is the fact that it can incorporate with your Subversion repository so that you can associate file revision check in with defects.

D. Flyspray

Flyspray is a web-based defect tracking system written in PHP. Flyspray is free software, released under the General Public License. This essentially means that you can get Flyspray and use it free of charge. The source code is available, and everyone is welcome to amend it to suit their needs. Its web pages describe it as “uncomplicated”, and the list of features includes: multiple database support (currently MySQL and PGSQL), multiple projects, 'watching' tasks, with notification of changes (via email or Jabber), comprehensive task history, CSS theming, file attachments, advanced search features, RSS/Atom feeds, wiki and plaintext input, voting, dependency graphs.

E. Redmine

Redmine is a flexible web-based project management web solicitation. Written using Ruby on Rails framework, it is cross-platform and cross-database. Redmine is open source and released under the terms of the GNU General Public License. Redmine is flexible issue tracking system. You can define your own statuses and issue types. He supports multiple projects and subprojects. Each user can have a different role on each project. Interface is very simple, intuitive and easy to navigate. Shortly, this is very good product and our recommendation.

F. Bug-Track

Bug-Track is web-based defect and bug tracking software permits you to document manage and assign all of your defects and tasks and allows you to organize your bugs, defects or issues into distinct projects. It can run on any web-server like Microsoft, Linux, UNIX, etc. Since it is a commercial application it is expected that it is better than other free products. But it isn’t true. He has nothing new and improved than other free bug tracking systems. One better thing is fact that he has more spontaneous interface than others and that is his only advantage.

G. Bugzero

Bugzero is a web-based bug, defect, issue and incident tracking software. Its single code based application which supports Windows and UNIX (based on Java™) and supports database systems as well as Access, MySQL, SQL Server, Oracle, and etc. Bugzero can be modified for software bug tracking, hardware defect tracking, and help desk customer support issue and incident tracking. Bugzero have intuitive interface but he lacks form features. The main disadvantage is the fact that Bugzero is a commercial product and you can find much improved product for free.

From all above presented, we conclude that, among the existing solutions, no one of them can be treated as the best one, for the general solution of this research. Each of them has some advantages and disadvantages. Some of them have some feature more than others but in the general, the set of features are the identical..

III. PROPOSED WORK

A. Module 1:- Admin

Defect management is crucial to closing the loop between requirements, implementation and verification and validation. Traditional defect tracking management, implemented in a standalone method, can no longer address the complexity and pace of change in modern software development. Defect management processes must be strongly interlinked with all of the other software development processes. The defect management process contains the following elements:

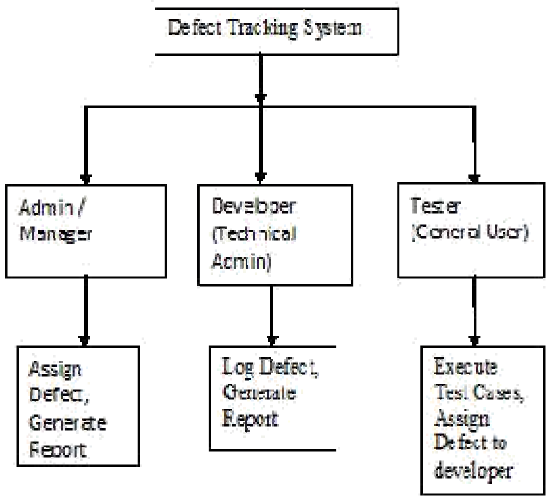
Defect Tracking System
Admin / Manager
Developer (Technical Admin)
Tester (General User)
Assign Defect, Generate Report
Log Defect, Generate Report
Execute Test Cases, Assign Defect to developer

Fig. 1. Process Flow Diagram.

In Defect Discovery it recognizes and report the potential defects. It ensures that the minimum essential information is captured. The information captured here should be advantageous to determine root cause and influence [9].

In Defect Examination & Prioritization, The team determines if the defect report is identical to an actual defect, if the defect has already been occurred, then find out the effect and priority of the defect is. Prioritization and scheduling is part of software development application and defect resolution is often part of it.

In Resolving Defect, the development team determines the origin of cause; implements the changes needed to fix the defect, and documents the details of the resolution in the defect management software, including ideas on how to judge the defect is fixed.

In Verification of Defects, The build holding the resolution to the defect is recognized, and testing of the build is performed to ensure the defect truly has been fixed.

The Admin achieve automatic generation of defect metrics for management reporting and process development purposes, as well as visibility into the presence and status of defects across all disciplines of the software development team.

Admin considering the following challenges such as:

(i) Detection and Notification

(ii) Defect Determination across Multiple Lines of Development

(iii) Automatic Defect Verification [4][6].

B. Module 2: - Developer

There are many ways in which we can categorize. Below are some of the classifications of Defects:

Severity Wise:

Major: A defect, which will cause a noticeable product failure or departure from requirements.

Minor: A defect that will not cause a failure in implementation of the product.

Fatal: A defect that will cause the system to crash or close shortly or influence other solicitations.

Work product wise:

SSD: A bug from System Study document

FSD: A bug from Functional Specification document

ADS: A bug from Architectural Design Document DDS: A bug from Detailed Design document Source code: A bug from Source code

Test Plan/ Test Cases: A bug from Test Plan/ Test Cases User Documentation: A bug from User manuals, operating Manuals

Status Wise:

- Open
- -Closed
- -Deferred
- -Cancelled

These are the major ways in which defects can be classified. [4], [9]

C. Module 3:- Tester

The step in defect life cycle varies from company to company. But the basic flow remains the same. However, below I'm describing a basic flow for Bug Life Cycle:

A Tester finds a bug. Status --> Open

Test lead review the bug and authorize the bug. Status --> Open

Development team lead reviews the defect. Status --> Open

The defect can be authorized or unauthorized by the development team. (Here the status of the defect / bug will be Open (For Authorized Defects) & Reject (For Unauthorized Defects).

Now, the sanctioned bugs will get fixed or deferred by the development team. Status of the fixed bugs will be Fixed & Status will be postponed for the bugs which got deferred.

The Fixed defects will be again re-tested by the testing team (Here based on the Closure of the defects, the status will be made as Closed or if the bug still remains, it will be re-raised and status will be Re-opened [10].

The above-mentioned cycle continues until all defects get fixed in the application.

The purpose of defect prevention is to identify the defects and take corrective action to ensure they are not repetitive over subsequent iterative cycles. Defect avoidance can be implemented by preparing an action plan to decrease or eliminate defects, generate defect metrics, defining corrective Action and producing an analysis of the origin causes of the defects [5].

Defect prevention can be accomplished by the following steps:

(i) Analyze defect data with periodic review using test logs from the execution phase: this data should be used to separate and categorize defects by root causes. This produces defect metrics highlighting the most creative problem areas.

(ii) Identify development strategies.

(iii) Escalate issues to senior administration or customer where essential.

(iv) Draw up an action plan to address exceptional defects and improve development process. This should be reviewed frequently for effectiveness and modified should it prove to be ineffective.

(v) Undertake periodic peer reviews to verify that the action plans are being adhered to.

(vi) Generate regular reports on defects by age. If the defect age for a particular defect is high and the severity is sufficient to cause concern, focused action needs to be taken to resolution.

(vii) Categorize defects into: critical defects, functional defects, and cosmetic defects.

The Track Defects sub process is designed to collect the data required to calculate and monitor the quality of the application, and also to control project risk and scope. The process is considered so that those with the best understanding of the customer priorities are in control of defect prioritization. The business analyst monitors a list of newly discovered issues using a defect tracking system like the Siebel excellence module. These users monitor, prioritize, and target defects with regular frequency. This is typically done daily in the early stages of a project, and perhaps several times a day in later stages.

The level of inspection is escalated for defects discovered after the project freeze date. A very careful dimension of the impact to the business of a defect versus the risk associated with introducing a late change must be made at the project level. Generally, projects that do not have appropriate levels of change management in place have difficulty reaching a level of system stability sufficient for deployment. Each change introduced carries with it some amount of regression risk. Late in a project, it is the dependability of the entire project team, including the business unit, to carefully manage the amount of change introduced.

Once a defect has been accepted to be fixed, it is assigned to development and a fix is designed, implemented, unit tested, and checked in. The testing team must then verify the fix by bringing the affected components back to the same testing phase where the defect was found. This needs regression testing (re execution of test cases from earlier phases). The defect is finally closed and verified when the component or module successfully passes the test cases in which it was exposed. The process of validating a fix can often have need of the re execution of past test cases, so this is one activity where automated testing tools can afford significant savings. One best practice is to define regression suites of test cases that allow the team to reexecute a relevant, comprehensive set of test cases when a fix is checked in. Tracking defects also collect the data required to measure and monitor system quality. Essential data inputs to the deployment readiness decision include the number of open defects and defect discovery rate. Also, it is important for the business customer to understand and grant the known open defects prior to system deployment [2], [4], [12].

In a peer review, co-workers of a person who created software work product examine that product to identify defects and correct shortcomings.

Verifies whether the work product correctly satisfy the specifications found in any predecessor work product, such as requirements or design documents

Identify any deviation from standards, including issues that may affect maintainability of the software.

Promote the exchange of techniques and education of the participant. All temporary and final development work products are candidates for review, including:

-Requirements specifications

-user interface specifications and designs

-architecture, high-level design, and detailed designs and models

-source code

-test plans, designs, cases, and procedures

-software development plans, including project management plan, configuration management plan, and quality assurance plan [5].

The principle of a defect tracking workflow is to move issues to resolution. When a defect is reported, system may require the following to occur:

(i) Verify the defect. Is it really a defect? Is it reproducible?

(ii) Assign resources to fix the defect. How much time and cost will development and QA need?

(iii) Release the fixed defect. When will it be released? Who approve releasing the change into a build? How are code change moved into new builds?

These questions, which affect project management, software development, QA, and release management, can all be forced through a defect tracking organization. For example, when a defect is added, it must be reviewed by a QA team member to ensure its correctness and authenticity. Once QA determine that a defect exists, a project manager must prioritize and then assign the defect to a developer to fix. After the defect is fixed, QA must test and validate the fix. A build manager must then ensure the fixed defect is released to the next build. A customer may even perform customer acceptance on the issue and verify the fix. Finally, the defect is closed after the fix is verified in the latest build or release [3], [4].

Defect Status:

- New: Default status when bug is reported
- Open: Indicates bug is assigned to review
- Reopen: Indicates testing team reopened the defect which was closed earlier
- Fixed: Indicates bug is verified
- Closed: Bug is closed and waiting for authorization by tester
- Rejected: Bug is rejected, rational for rejecting defect to be provided [4], [11].

IV. INPUT OUTPUT SPECIFICATION AND DISCUSSION

Testing is an integral part of any system or project. The various objectives of Testing to obtain the expected results:

(i) To uncover the errors in function logic

(ii) To verify that software needs the specific requirement.

To verify that software has been implemented according to the predefined standards

Category no.	Type of Tracking	Actual Results
1	Report generation	Systems keeps track of the defect and generates report
2	Email Notification	Sends an email to the individual who created defect along with the module name

V. CONCLUSION

Defect Tracking System is very useful for removing defects from project module. Only if features mentioned in document are extended. In future users may also be possibly notifying the name/Id of defect creator so that organization can prevent their systems.

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

(i) It's a web-enabled scheme.

(ii) This task offers user to enter the data through simple and interactive forms. This is very useful for the client to enter the desired information through so much simplicity.

(iii) The user is mainly more worried about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or update so that the user cannot enter the invalid data, which can build problems at later date.

(iv) Sometimes the user finds bug in the later stages of using Project that he needs to update some of the Information that he entered earlier. There are options for him by which he can update the records. Moreover there is limitation for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.

Current Defect tracking systems do not effectively elicit all of the information needed by developers. Without this information developers cannot resolve defects in a timely fashion and so we consider that improvements to the way defect tracking systems collect information are essential. This is likely to speed up the process of resolving bugs. In the future, I will move from the current prototype of the interactive system to a full-scale system that can deal with a variety of information to gather, as generally observed in the real world. [7], [10].

REFERENCES

[1].BasicInspectionprocess, http://www.cs.toronto.edu/~sme/CSC444F/hand outs/inspection_process_model.pdf

[2]. Gao, Kehan, et al. "Choosing software metrics for defect prediction: an investigation on feature selection techniques." Software: Practice and Experience 41.**5** (2011): 579-606.

[3].Introduction,http://www.mks.com/solutions/disc ipline/dm/defectmanage ment?gclid

[4]. Jalote, Pankaj, and Naresh Agrawal. "Using defect analysis feedback for improving quality and productivity in iterative software development." Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on. IEEE, 2005.

[5]. Kalinowski, Marcos, David N. Card, and Guilherme H. Travassos. "Evidence-based guidelines to defect causal analysis." Software, IEEE 29.**4** (2012): 16-18.

[6]. Kocher, Paul, et al. "Introduction to differential power analysis." *Journal of Cryptographic Engineering* 1 (2011): 5-27.

[7]. Kumaresh, Sakthi, and R. Baskaran. "Defect analysis and prevention for software process quality improvement". *International Journal of Computer Applications* (0975–8887) Volume (2010).

[8]. Lauesen, Soren, and Otto Vinter. "Preventing requirement defects: An experiment in process improvement" Requirements Engineering 6.1 (2001): 37-50.

[9]. Life Cycle, http://www.softwaretestingstuff.com/2008/05/bug-life-cycle.html.

[10]. Maintenance and Environment, Conclusion http://www.onestoptesting.com/test-cases/defect-tracking.asp

[11]. R.B. Lenin & R.B. Govindan," Predicting Bugs in Distributed Large Scale Software Systems Development", 2008.

[12]. Stephen Blair, "A Guide to Evaluating a Bug Tracking System", October, 2004.

[13].Trajkov Marko & Smiljkovic Aleksandar, "A Survey of Bug Tracking Tools", 2006.