SOFTWARE DEFECT MANAGEMENT PROCESS RATING

Kadatskaja O., Ghanem Karim

V.V. Popovsky Dep.Engineering Infocommunication HNURE,
Ukraine

E-mail: olha.kadatska@nure.ua,

Abstract

Defects are destructive at all stages of software development, need use the defect management process keep their reduced to a minimum and to deal with potential defects. Identifying the root causes of defects requires a systematic approach to analyzing the problem. To effectively control and deal with defects, it is necessary to manage their life cycle properly. There are different approaches to root cause analysis each of these methods has its own advantages and disadvantages, and the choice of method depends on the type of problem and the resources available.

Defect/bug life cycle representant of the various states of a defect in which it resides from the initial to the final stage of its existence and can be customized to the processes of a specific project. The structure of software diagnostics occurs at two main levels - analysis of source and binary (executable) codes, each level of development has its own advantages and achievements. However, defects generally result in software developers being unable to take into account all aspects of the program's executable code that arise from the use of optimizing compilers and binary translators.

Software defects you be divided into vulnerabilities (critical errors) leading to malfunction, denial of service, changes in the security of information resources and errors (non-critical) that only affect the quality of the software system (for example the program uses more memory to operate than necessary due to memory leaks when working with it). The search for critical errors is of greatest interest, however, non-critical errors also affect the process of software operation, therefore modern diagnostic systems are aimed at identifying both types of defects.

The current generally accepted classification of defects by type is presented in the Common Weakness Enumeration database, the list of registered defects is in the Common Vulnerabilities and Exposures database of the MITER organization [1].

Defect management occurs systematically during the process of identifying and eliminating errors. The defect management cycle can be dividing into the following stages: defect discovery, defect categorization, defect resolution by developers, review by testers, defect closure, defect reporting at the end of the project. Moreover, defects have different statuses, namely: new - if a new defect is registered and published for the first time; assigned-publishing a bug by a tester, the tester's manager approves the bug and transfers it to the development team. Open status - the developer begins analysis and works to fix the bug; fixed - the developer made the necessary change to the code and checked it; pending retest - once the defect is fixed, the developer provides the specific code to the tester to retest the code. Retest status- at this stage, the tester retests the code to check whether the developer has fixed the defect, verified - the tester retests the bug after it has been fixed by the developer. Reopened status, if the bug persists even after the developer has fixed the bug. And again, the bug goes through its life cycle. Closed - if the bug no longer exists, duplicate - if the defect is repeated twice or the defect corresponds to the same error concept. Statuses rejected - if the developer believes that the defect is not such, deferred - if the current bug is not a priority and is expected to be fixed in the next release, not a bug - if it does not affect the functionality of the application.

Critical defects pose a security risk to users, the defect must be corrected immediately otherwise it may lead to large losses for the software. Testing only reduces the likelihood of defects found in the software, but does not guarantee their absence, exhaustive testing is impossible. Complete testing using all combinations of input data, results, and preconditions is physical-ly impossible.

Most of the defects are found in a limited number of modules. The pesticide paradox emerges if you repeat the same test cases over and over again, at some point this test suite will stop identifying new defects. Testing is done differently depending on the context. The absence of defects found during testing does not always mean the product is ready for re-lease. The system must be user-friendly to use and meet his expectations and needs - quality assurance and quality control - analysis of test results and the quality of new versions of a released product.

Following management tools should be used to evaluate software defect management. Checklist in testing is management tool based on the principles of "error protection"; it is a sheet with a list of necessary checks and notes on their implementation. To compile a checklist, you first need to think through defect-dangerous areas. Universal checklists have the same wording, they are prepared for checking identical (similar) objects without reference to any specifics of a particular software. Specialize checklists are developed for the software under test, tied to the unique requirements/features of this software. Software for creating checklists - Testpad, Checklists. expert, Notion, Evernote, etc.

Test case is a step-by-step description of the actions that need to be performed to test any software function; this is an algorithm that a tester must follow (model user behavior) in order to check the functionality of a certain part of the code. As a rule, test cases are written for repeated testing basic functions, the functionality of which must be verified every time the software is updated, for example, the authorization function.

To localize defects, it is necessary to identify the causes of the defect, analyze the possibility of the influence of the found defect on other areas, find deviations from the expected result, and explore the environment - reproduce the bug in different operating systems (Android, Windows, etc.) and browsers (Google Chrome, Internet Explorer, etc.).Also necessary check the bug on different devices, check it in different software versions and analyze system resources. Evidence of bug reproduction should be recorded using logs, screenshots, or screen recordings.

All detected defects must be filed as bug reports for all project specialists to reproduce the detected defect and understand its criticality.

Identifying the root causes of defects requires a systematic approach to analyzing the problem. There are different approaches to root cause analysis such as 5 whys, Ishikawa diagrams, Pareto dicars. Each of these methods has its own advantages and disadvantages, and the choice of method depends on the type of problem and the resources available. 5 whys simple but effective tool for identifying the root cause of a problem involves asking "why" five times to get to the root cause of the defect. Ishikawa diagrams are a visual tool for identifying the causes of defects, helping to identify primary and secondary causes and how they relate to each other. Pareto dicars are useful for identifying the most significant causes of defects, helping to prioritize the causes and focus on the most important ones.

Once the defects are accepted and classified, the following steps must be followed to correct them. The programmer eliminates the causes of the defects to change the status to meet the requirements. The developer side implements a schedule for eliminating these defects depending on their priority. The test manager monitors the progress of the bug fixes based on the schedule and generates a bug fix report from the developers when the defects are fixed. After the development team has fixed the defect and re-ported it, the testing team verifies that all reported defects have actually been fixed and their status is changed to closed. If the defect is not resolved, a notification is sent to the development department to check the defect again, and then a report is generated. The quality of the test is assessed by the following parameters - defect reject ratio (DRR) and defect leakage ratio (DLR) and the lower the DRR and DLR value, the better the quality of testing.

Conclusion

Analyzing software defect management process allows us to detecting errors with a well-planned and controlled defect life cycle which shown of how developers wrote the code and whether testers did their job correctly. Based on analysis of tools we chouses corresponding with defect management cycle and categorization defects status. Is proposed strategy for preventing software security issues and ways must be followed to correct defects.

References

1. Protecting Against Malicious Use of Remote Monitoring and Management Software.. URL:https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-025a (02, 2023)