UDK 004.414.3

Svitlana Volkova

CLEAR DESCRIPTION FOR SOFTWARE DEFECT'S LIFECYCLE

Petro Mohyla State Humanities University in consortium

with "Kiev-Mohyla Academy", Mykolaiv, Ukraine

Comprehensible description for defect's lifecycle with additional explanation is presented. The relation among software defect, error and failure is described. The importance of software testing in providing safety-critical system's quality is determined.

Key words: software development life cycle, defect, error, failure, software quality, software testing, metrics, defect's status.

Introduction

In spite of extreme development of modern methodologies in information technologies (IT) sphere a human being can make errors during software development life cycle, which produces **defects**[1] in the system's code. If a defect in code is executed, the system will **fail**[2] to do what it should do (or do something it should not), causing a **failure**[3]. A fault can also turn into a failure when the environment is changed [1]. The relation among software's defect, fault and failure is presented in Fig.1.

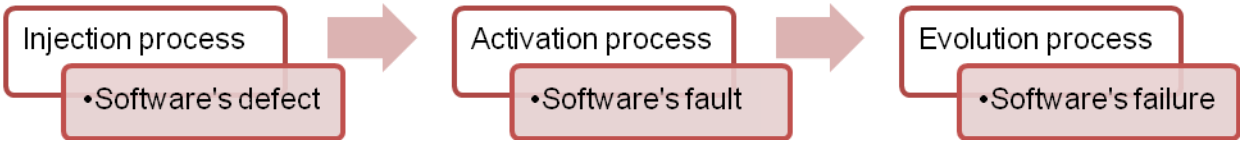

Fig.1.The relation between software's defect, fault and failure

Problem statement

In accordance to no-failure operation of critical systems the essential issue about high-quality software development exists. For basic resolution of this issue we can use **software testing**[4], that is defined as basic method of software quality control and estimation. The relation among **software quality**[5] and testing specification is described in international standard ISO 9126:2001 [2-6].

There are a number of common software measures, often called **metrics**[6], which are used to measure the quality of the software or the adequacy of the testing [7-10]:

defects found per tester per unit time (Day/Week/Month); total defects found in a release; total defects found in a module/feature; defects found/fixed per build; number of customer reported defects, as a measure of testing effectiveness; defects trend over the period in a release, because defects should converge towards zero as the project gets closer to release etc.

Above mentioned software testing metrics define the essential additional task's resolution for general of description the **defect's status**[7] during software testing stage. The necessity of basic defect status's explanation determines by different approaches of existed bug-tracking system to defect's status description. That's why the general way of defect's lifecycle with essential explanation is presented in Fig.2.

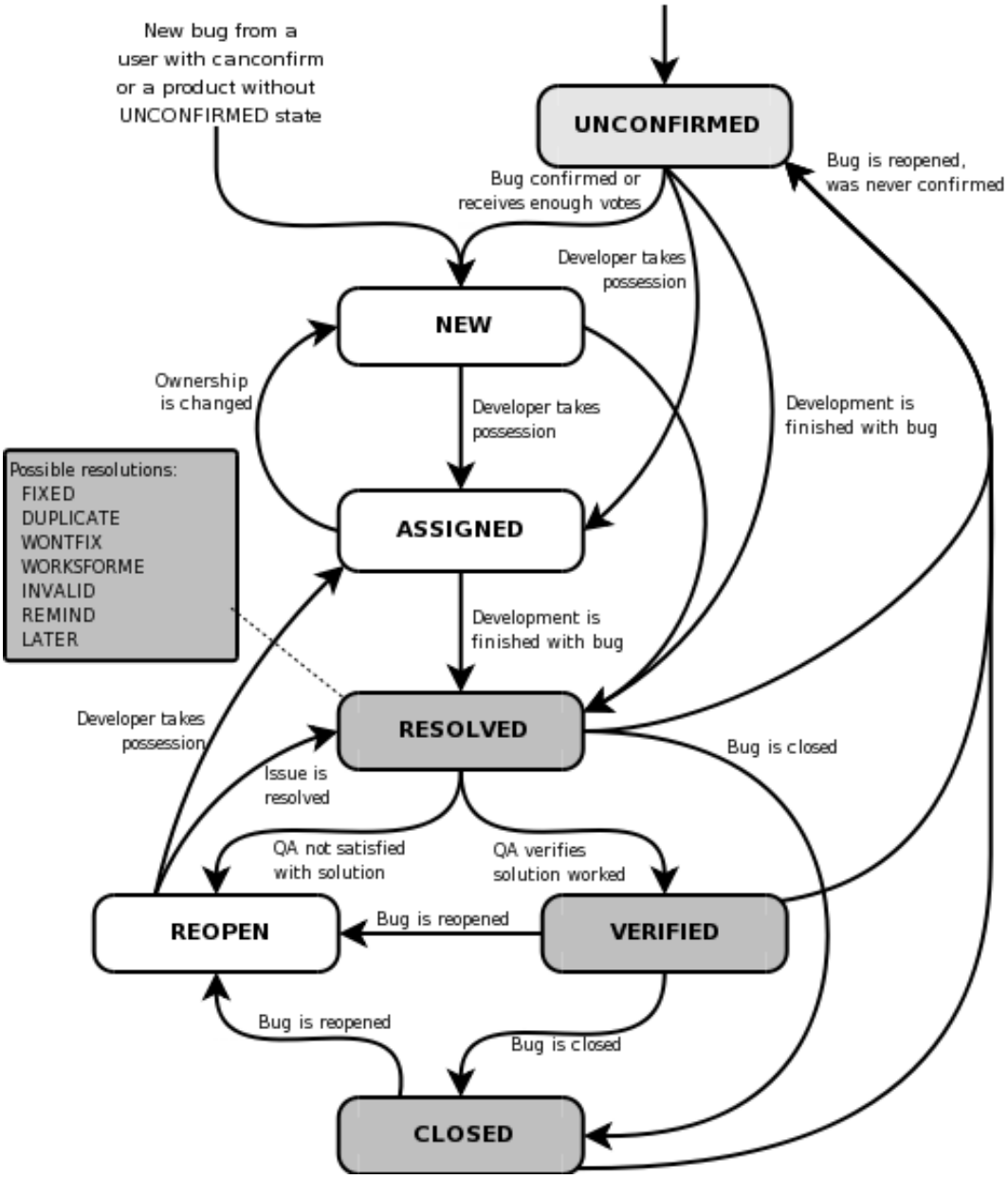

Fig.2.General description of defect's lifecycle

Defect becomes with status of NEW in the following 2 cases: this is the first status of the defect after submitting it to bug-tracking system; ownership of the defect is changed.

Defect can receive the status of ASSIGNED from the following 2 states: defect can be reassigned to other person, in this case it receives status of NEW; defect can be RESOLVED with receiving the same status.

Defect may become RESOLVED from any of these 3 “not resolved” statuses: NEW; ASSIGNED; REOPENED.

From the status RESOLVED there are 3 possibilities: after verification by QA defect becomes VERIFIED; defect is directly CLOSED, omitting the verification; defect is REOPENED when resolution was deemed wrong.

RESOLVED status necessarily needs applying one of pre-defined possible resolutions to a defect: defect is and now fix needs verification by QA; INVALID resolution means that person responsible for resolving the issue considers it as not a defect at all; WONTFIX resolution means that person responsible for resolving the issue agrees that it is a defect, but it will never be fixed for some reasons; LATER resolution means that person responsible for resolving the issue agrees that it is a defect, which needs to be fixed, but it will not be fixed in this version of the product; REMIND resolution means that person responsible for resolving the issue agrees that it is a defect, which needs to be fixed, but it will probably not be fixed in this version of the product for some reasons; DUPLICATE resolution means that person responsible for resolving the issue considers it as a duplicate of an existing defect; WORKSFORME resolution means that person responsible for resolving the issue was unable to reproduce the defect.

Defect can be marked as VERIFIED only from the status RESOLVED. From the status VERIFIED there are 2 possibilities: defect is CLOSED, this usually happens when the product under test is released; defect is REOPENED when resolution was deemed wrong.

Defect can be marked as CLOSED from any of 2 other “resolved” statuses: RESOLVED or VERIFIED.

Conclusions

As a result, defect can only exist in one of the statuses below. Firstly, defect should not remain unresolved or not verified for a long time. Secondly, defect should be resolved and verified in order of their priority that determines the correctness of software testing and for this reason defines the software quality.

Consequently, clear and understandable description of software defect's status during software life cycle leads to increasing of software quality, that especially important for safety-critical systems.

Glossary

[1]**Software defect** is obvious or hypothetical cause of fall over, that means deviation of expected results for correct user's service. [2]**Software error** is software code's or documentation's record, that can cause an incorrect result. [3]**Software failure** means the demonstration of software inoperativeness. [4]**Software testing** is observation for software functioning in specific conditions for the purpose of definition of conformity's degree to software requirement. [5]**Software quality** is dynamic characteristic that defines if product is defect free and if it meets or conforms to the statement of requirements that defines the product. [6]**Testing metrics** are numbers of common software measures, which are used to measure the software's state or the testing's adequacy. [7]**Defect's status** is the possible defect's condition in accordance to its effective correction during software life cycle.

References

1. The 10 greatest catastrophes caused by PC carelessness, webpage: http://tech.blorge.com/Structure:%20/2007/03/02/the-10-greatest-catastrophes-caused-by-pc-carelessness/
2. ISO/IEC 9126-2001, Software engineering – Product quality (Part 1 – 4).
3. SWEBOK – Guide to Software Engineering Base of Knowledge // IEEE Computer Society, 2004. http://www.swebok.org/
4. Cem Kaner, James Bach, & Bret Pettichord, Lessons Learned in Software Testing, John Wiley & Sons, 2002.
5. J. Whittaker, How to Break Software: A Practical Guide to Testing, 2002.