

# Software Testing

## Lecture (2)

Dr. Manar Elkady

[m.elkady@fci-cu.edu.eg](mailto:m.elkady@fci-cu.edu.eg)

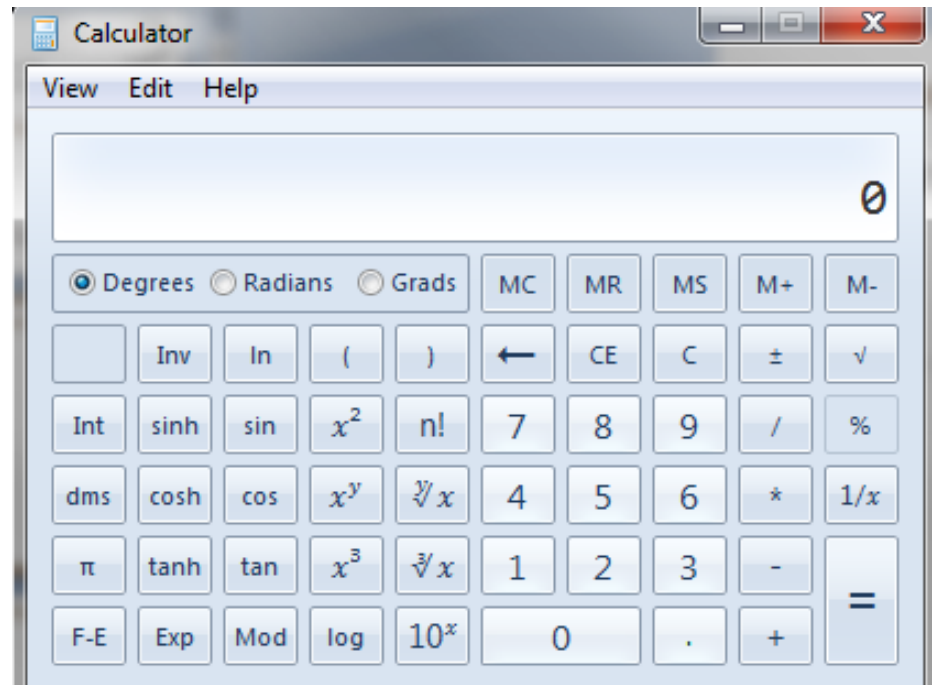
*These material are retrieved from previous courses offering by Dr. Soha Makady and Prof. Amr Kamel*

# Outline

- A Software bug
- An Effective Bug Report
- Severity and Priority
- A Bug's Lifecycle
- Bug/Defect Tracking Systems

# Our Calculator!

- A **product specification (spec.)** defines the product, what it will do, how it will act, and what it won't do.
- Lets consider our calculator product.
- Consider that you, as a tester, receive that product to test it.



## Our Calculator! (Cont'd)

- Should you consider the following a bug, or no?
  1. As a tester, you press the + key, and nothing happens.
  2. The spec. states that the calculator should never crash or freeze.
    - You pound on the keys, and you get the calculator to stop responding to your input.
  3. Besides addition, subtraction, multiplication, and division, the calculator **correctly** performs square roots.

## Our Calculator! (Cont'd)

- Should you consider the following a bug, or no?
  4. When the battery gets weak, you start getting wrong answers for your calculations.
    - The spec. ever considered how the calculator should react in such mode.
  5. You find the buttons too small. The display is difficult to read.
- ALL of the above ARE bugs.

# A Software Bug

- A bug occurs when one, or more of the following rules is true:

As a tester, you press the + key, and nothing happens.

- Rule 1: The software doesn't do something that the spec. says it should do.

The spec. states that the calculator should never crash or freeze.

- Rule 2: The software does something that the spec. says it shouldn't do.

# A Software Bug

- A bug occurs when one, or more of the following rules is true:

Besides addition, subtraction, multiplication, and division, the calculator correctly performs square roots.

- Rule 3: The software does something that the spec. does not mention.

When the battery gets weak, you start getting wrong answers for your calculations.

- Rule 4: The software doesn't do something that the specification does not mention but should.

# A Software Bug

- A bug occurs when one, or more of the following rules is true:

You find the buttons too small. The display is difficult to read.

- Rule 5: The software is difficult to understand, hard to use, slow ...etc. (i.e., something is just plain not right).



# A Software Bug

- Why would a bug be not fixed?
  - There isn't enough time.
  - It is not really a bug. It's a feature!
  - It is too risky to fix.
  - It is just not worth it.
  - Ineffective bug reporting.
    - “Whenever I type a bunch of random characters in the login box, the software starts to do weird stuff.”
    - Any comments?
- So... What should an effective bug report look like?

# An Effective Bug Report

- Minimal:
  - Eclipse's study with developers: "If people open rude or sarcastic bugs, it doesn't help their chances of getting their issues addressed."
- Singular:
  - "The following words are misspelled on 15 different pages in the online help file: ....". *What do you think of that report?*
  - "The login dialog won't accept passwords or login IDs with uppercase characters". *What do you think of that report?*

# An Effective Bug Report

- Reproducible.
  - E.g., Try to isolate what seems like a random behavior.
  - How?
- But ... Are all bugs equal?
  - No!

# Bugs Are NOT Equal

- Reported bugs get classifications to clarify their impact.
- Each bug gets assigned a **severity** and a **priority**.
- **Severity**: indicates how bad the bug is.
- Severity possible values:
  - **Sev. 1**: system crashes, security breach.
  - **Sev. 2**: wrong result, loss of functionality.
  - **Sev. 3**: Minor problem, misspelling, UI layout, rare occurrence.
  - **Sev. 4**: Suggestion.

## Bugs Are NOT Equal (Cont'd)

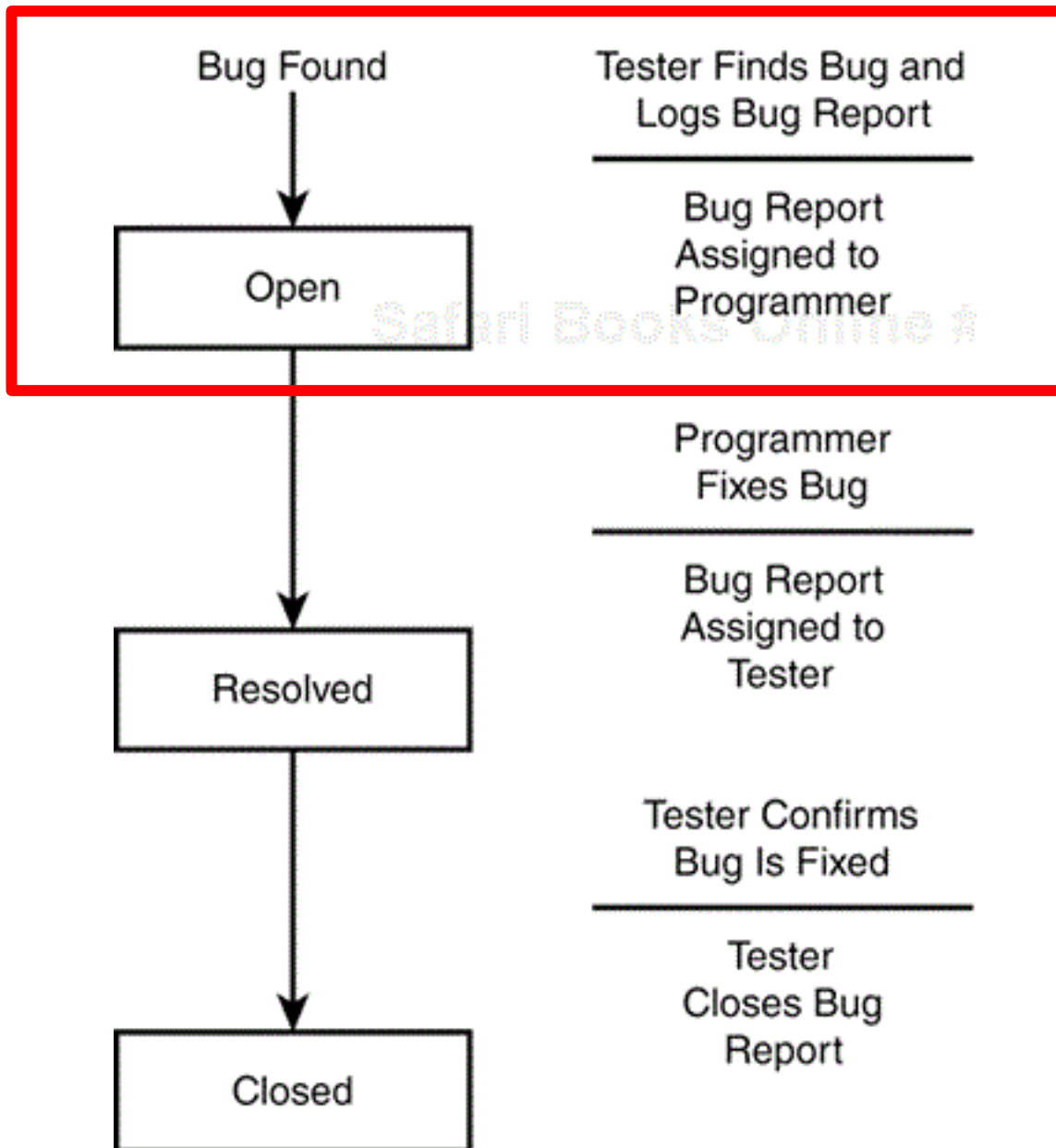
- **Priority:** indicates how much emphasis should be placed on fixing the bug.
- Priority possible values:
  1. **Immediate fix:** blocks further testing, very obvious.
  2. **Must fix** before the product is released.
  3. **Should fix** when the time permits.
  4. **Would like to fix** but the product can be released as is.
- **Let's see some examples.**

## Bugs Are NOT Equal (Cont'd)

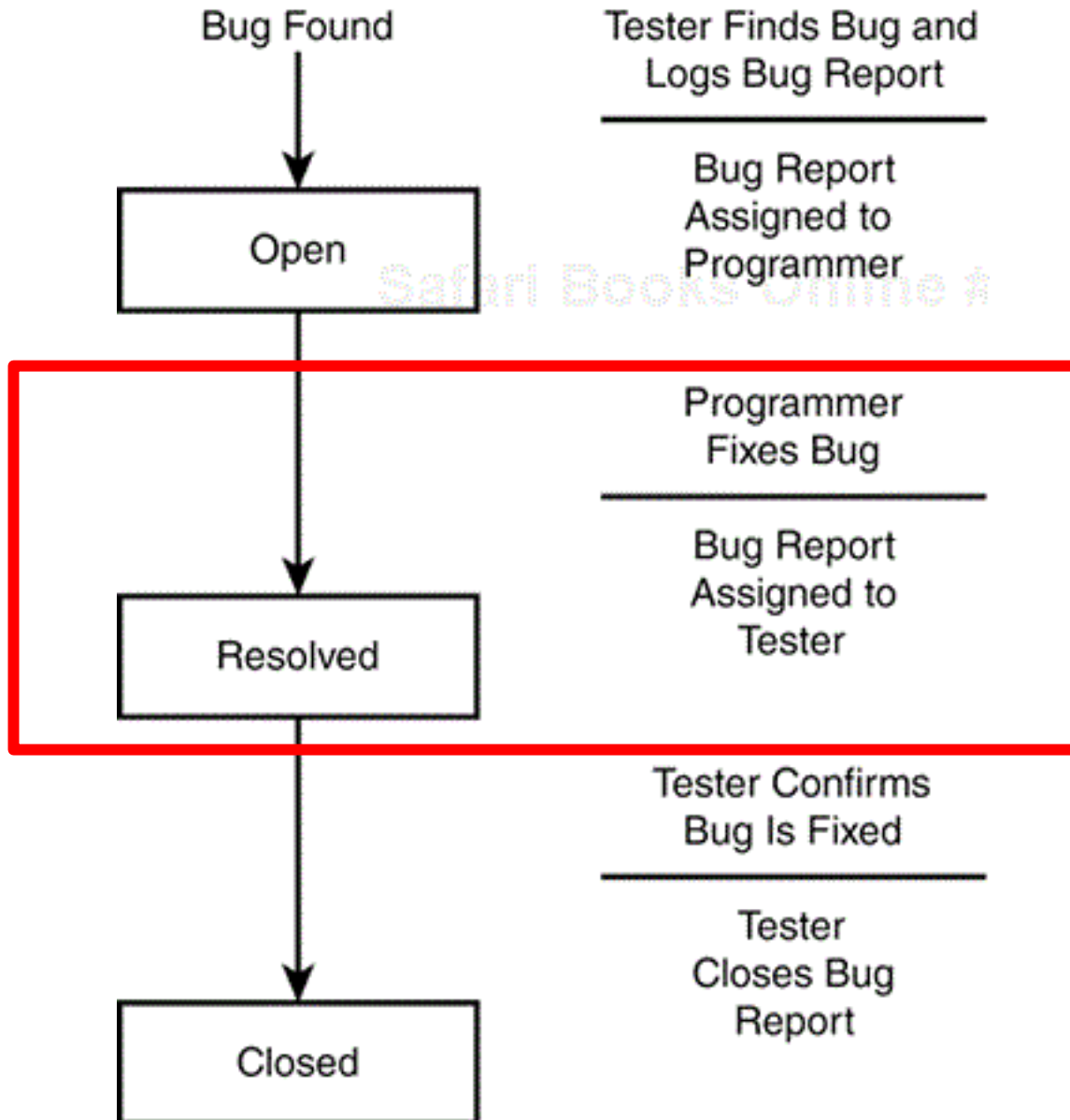
What are the severities/priorities of the bugs below?

- A. A data corruption bug that happens very rarely.
- B. A misspelling in the setup instructions that causes users to phone in for help.
- C. A software release (for testing) that crashes on startup
- D. A button should be moved a little bit to the left

# A Bug's Lifecycle

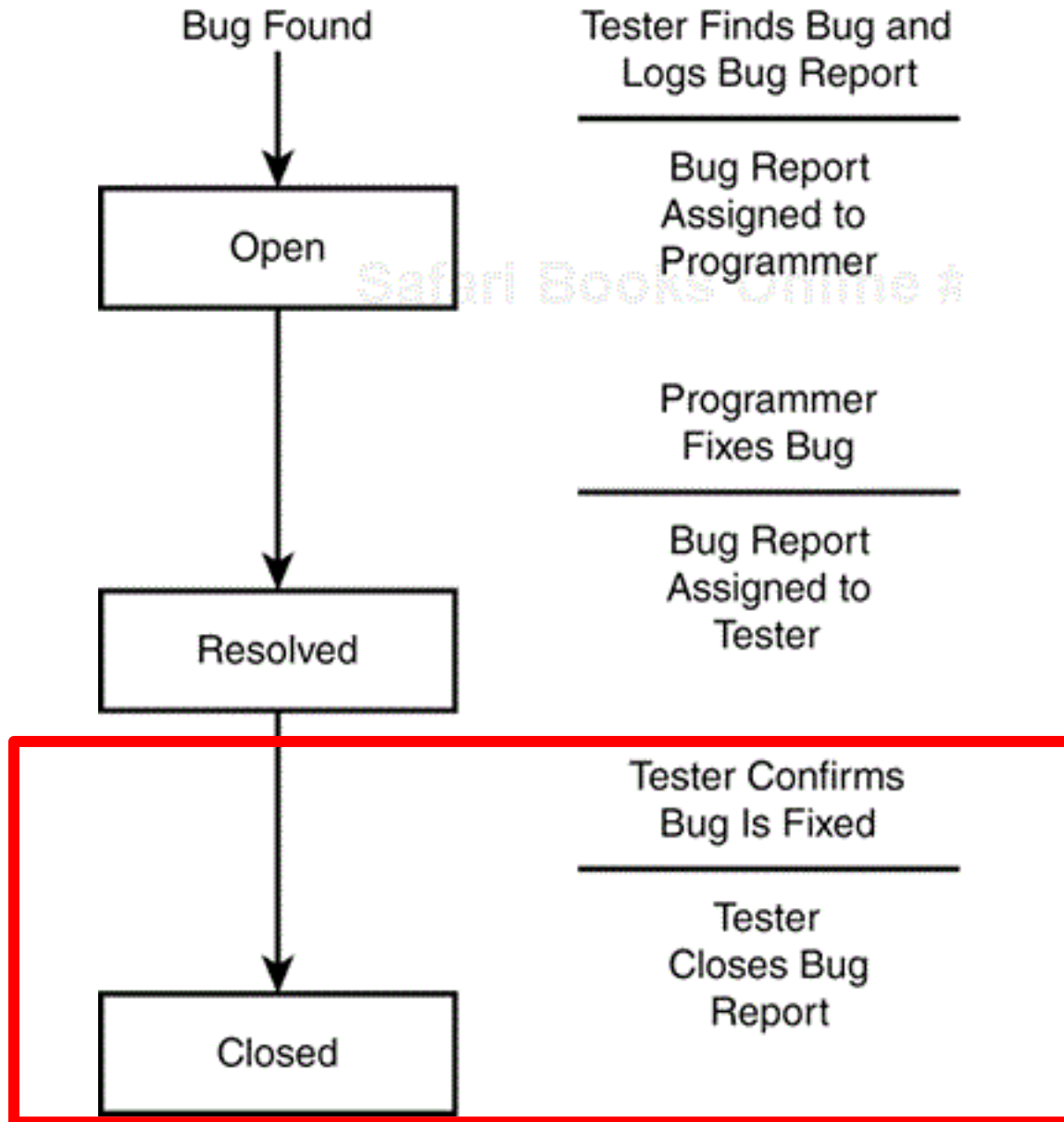


# A Bug's Lifecycle

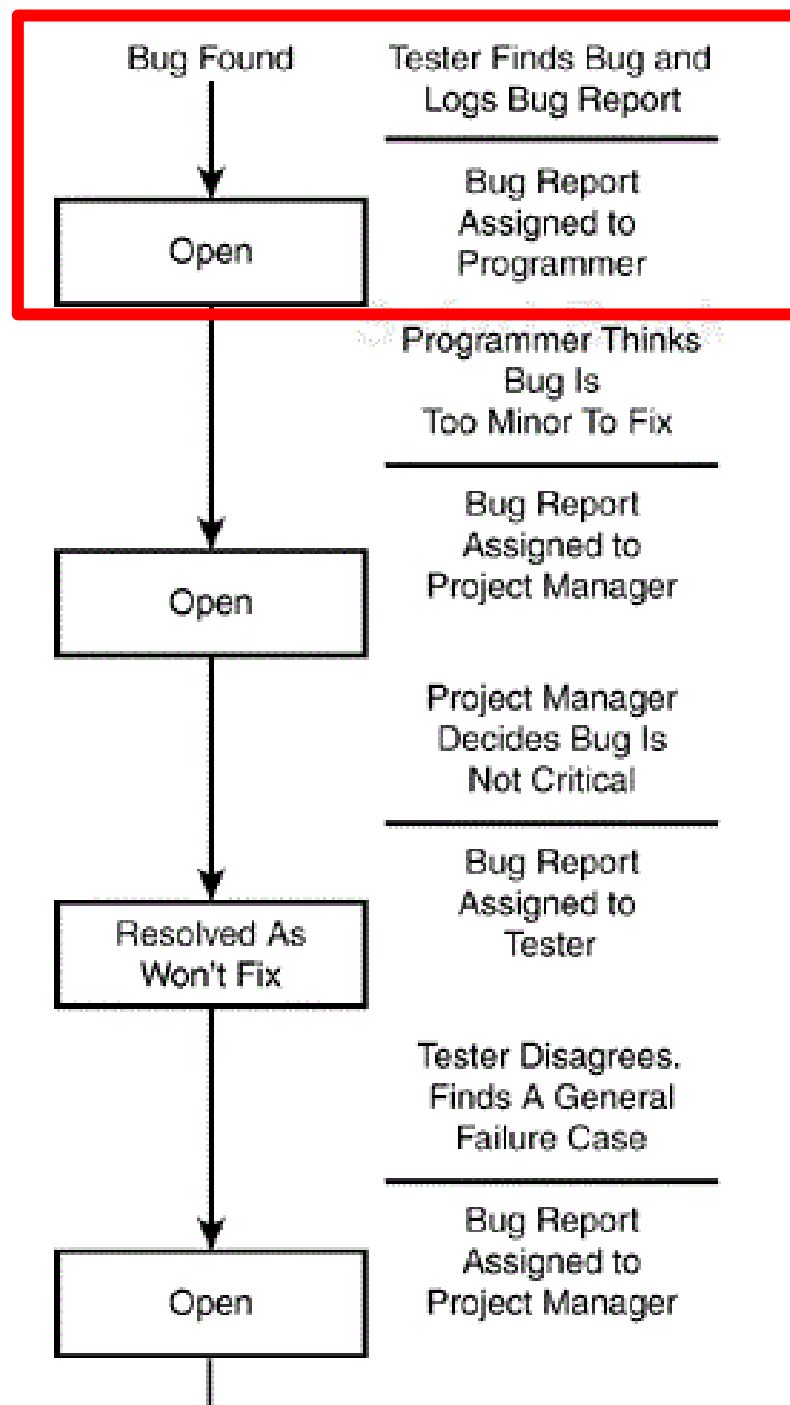




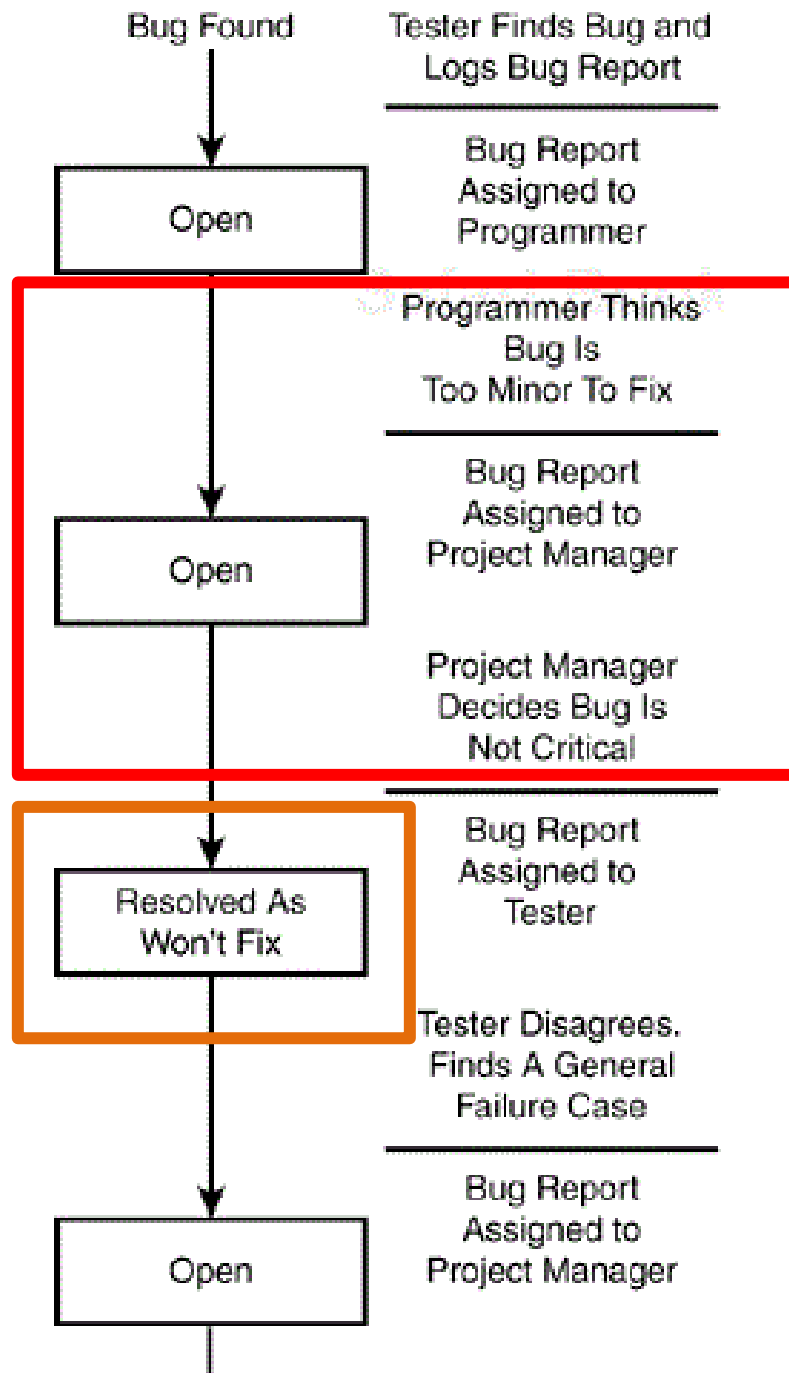
# A Bug's Lifecycle



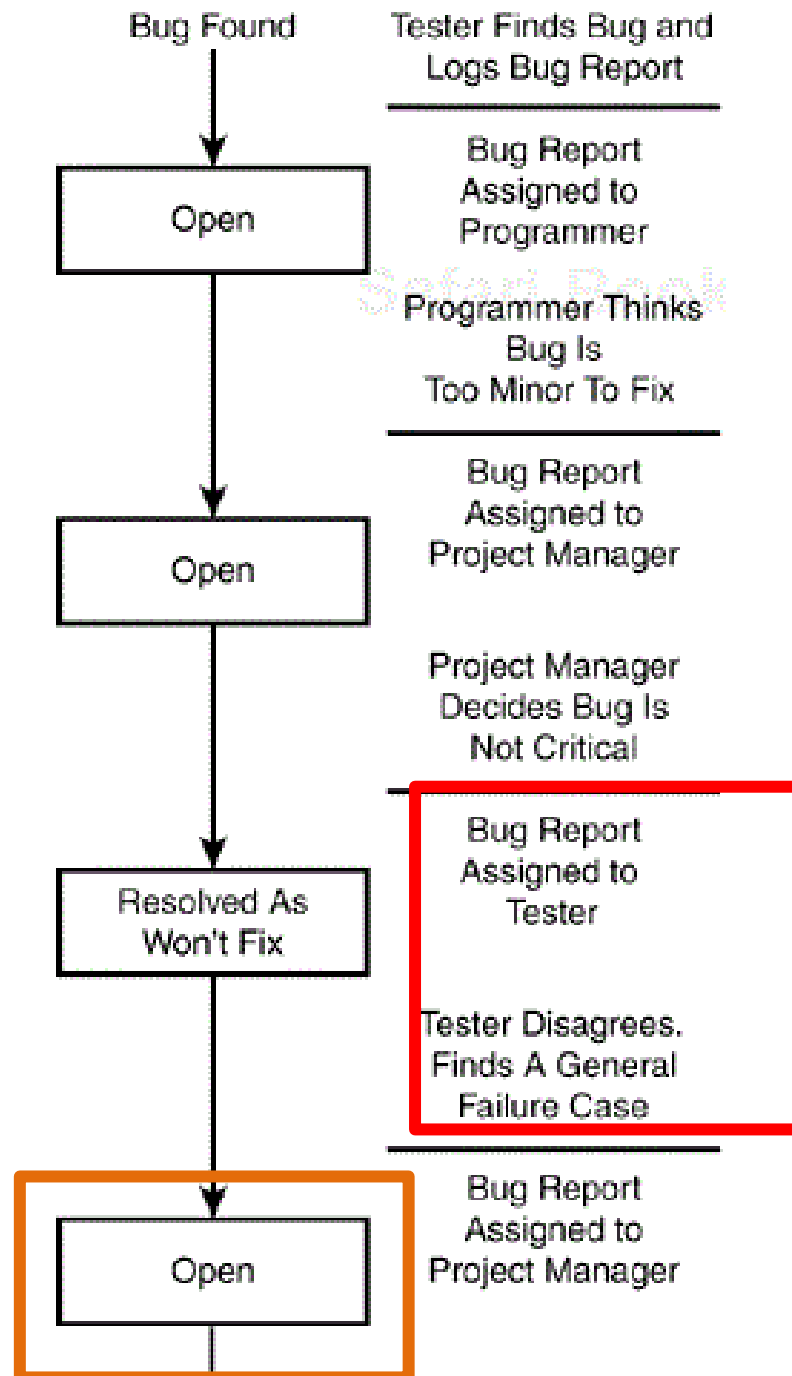
## A complicated Bug's Lifecycle



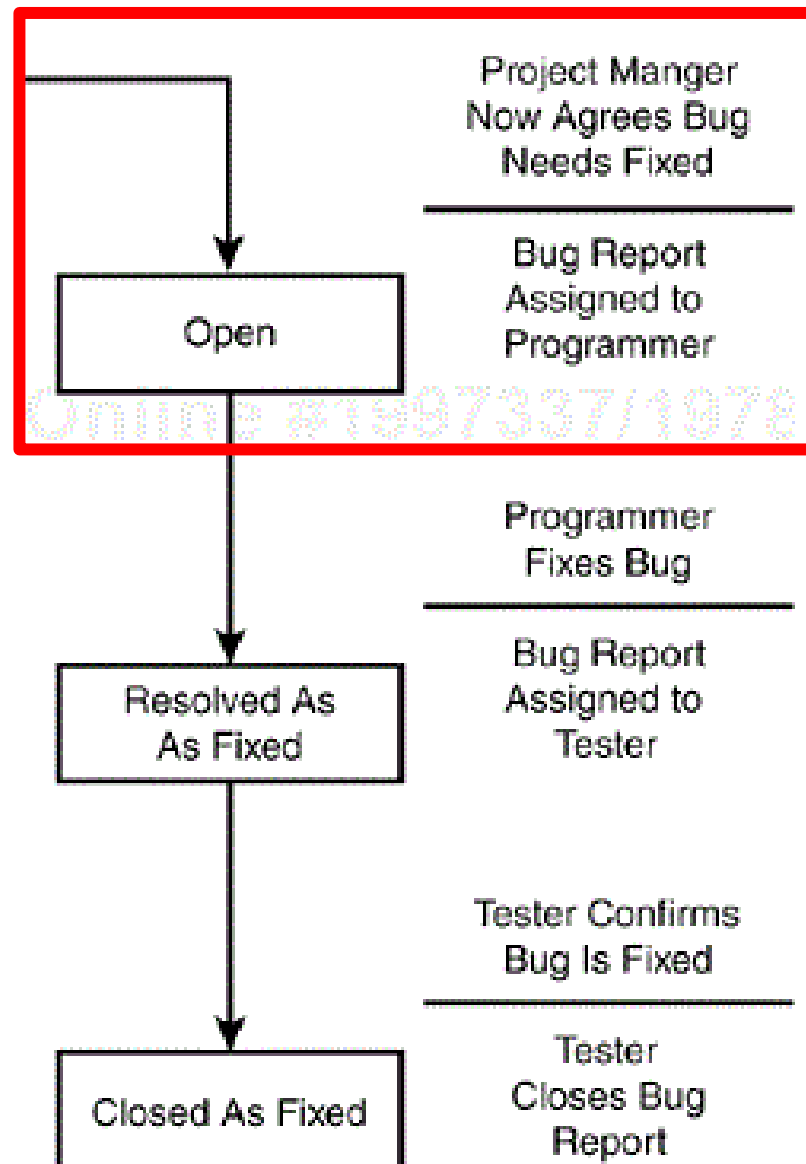
## A complicated Bug's Lifecycle



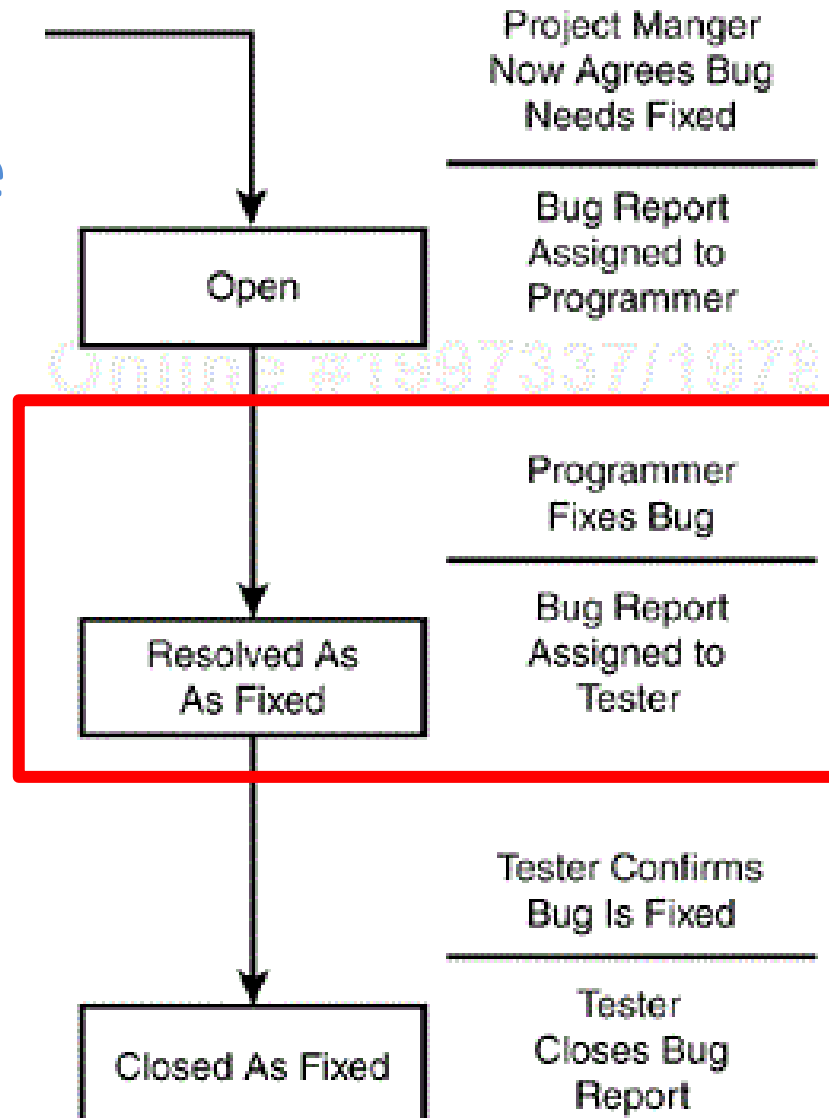
# A complicated Bug's Lifecycle



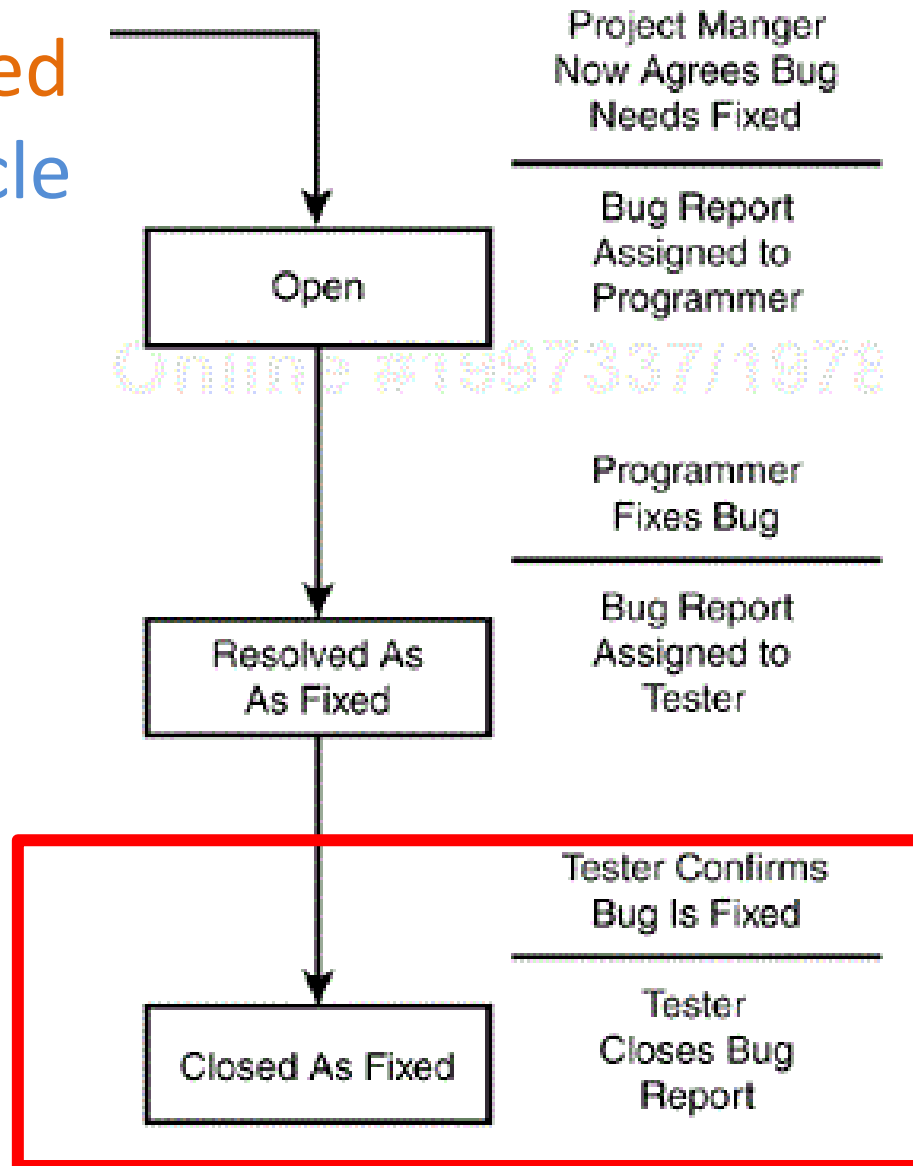
## A complicated Bug's Lifecycle



## A complicated Bug's Lifecycle



## A complicated Bug's Lifecycle



# Bug/Defect Tracking Systems

- A bug tracking system keeps track of reported bugs.

WIDGETS SOFTWARE INC.      **BUG REPORT**      BUG#: \_\_\_\_\_

SOFTWARE: \_\_\_\_\_ RELEASE: \_\_\_\_\_ VERSION: \_\_\_\_\_

TESTER: \_\_\_\_\_ DATE: \_\_\_\_\_ ASSIGNED TO: \_\_\_\_\_

SEVERITY: 1 2 3 4      PRIORITY: 1 2 3 4      REPRODUCIBLE:    Y    N

TITLE: \_\_\_\_\_

DESCRIPTION: Safari Books Online #199731

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_



# Bug/Defect Tracking Systems

- A bug tracking system keeps track of reported bugs.

RESOLUTION: FIXED DUPLICATE NO-REPRO CAN'T FIX DEFERRED WON'T FIX

DATE RESOLVED:\_\_\_\_\_RESOLVED BY:\_\_\_\_\_VERSION:\_\_\_\_\_

RESOLUTION COMMENT:\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

=====

RETESTED BY:\_\_\_\_\_VERSION TESTED:\_\_\_\_\_DATE TESTED:\_\_\_\_\_

RETEST COMMENT:\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

=====

# Real Examples

- Several automated bug tracking tools exist around
  - Bugzilla
  - SourceForge's bug tracking
  - Github issue tracking
  - CodePlex
- Let's see some real bugs
  - Eclipse IDE
  - [https://bugs.eclipse.org/bugs/show\\_bug.cgi?id=257699](https://bugs.eclipse.org/bugs/show_bug.cgi?id=257699)
  - [https://bugs.eclipse.org/bugs/show\\_bug.cgi?id=474525](https://bugs.eclipse.org/bugs/show_bug.cgi?id=474525)

## Required Readings

- Ron Patton. Software Testing, 2<sup>nd</sup> edition.
  - Chapter 18: Reporting What You Find

# References

- Ron Patton. 2005. *Software Testing (2nd Edition)*. Sams, Indianapolis, IN, USA.
- Glenford J. Myers, Corey Sandler, and Tom Badgett. 2011. *The Art of Software Testing* (3rd ed.). Wiley Publishing.



276305	212202.FCI.SCS357	Software Testing
--------	-------------------	------------------