Software Testing

Soha Makady

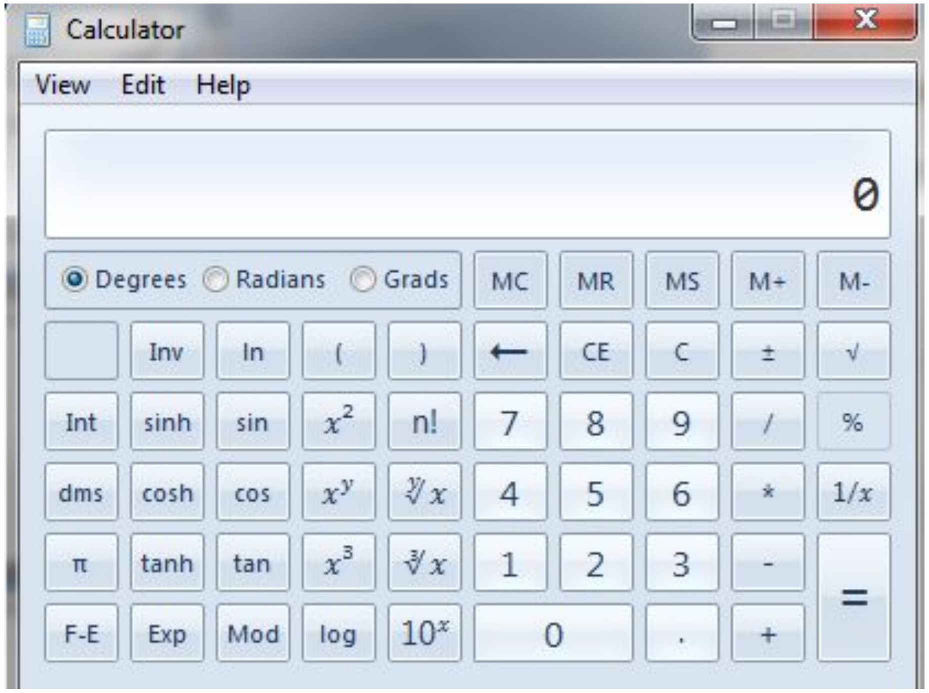
Some of the material are retrieved from a previous course offering by Prof. Amr Kamel

Outline

- A Software bug
- A Bug’s Lifecycle
- Software testing – Basic definitions
- Software Testing principles

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:
  - "In Eclipse, 205 bug reports were submitted for **Windows**, but later re-assigned to **Linux**".
- 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)

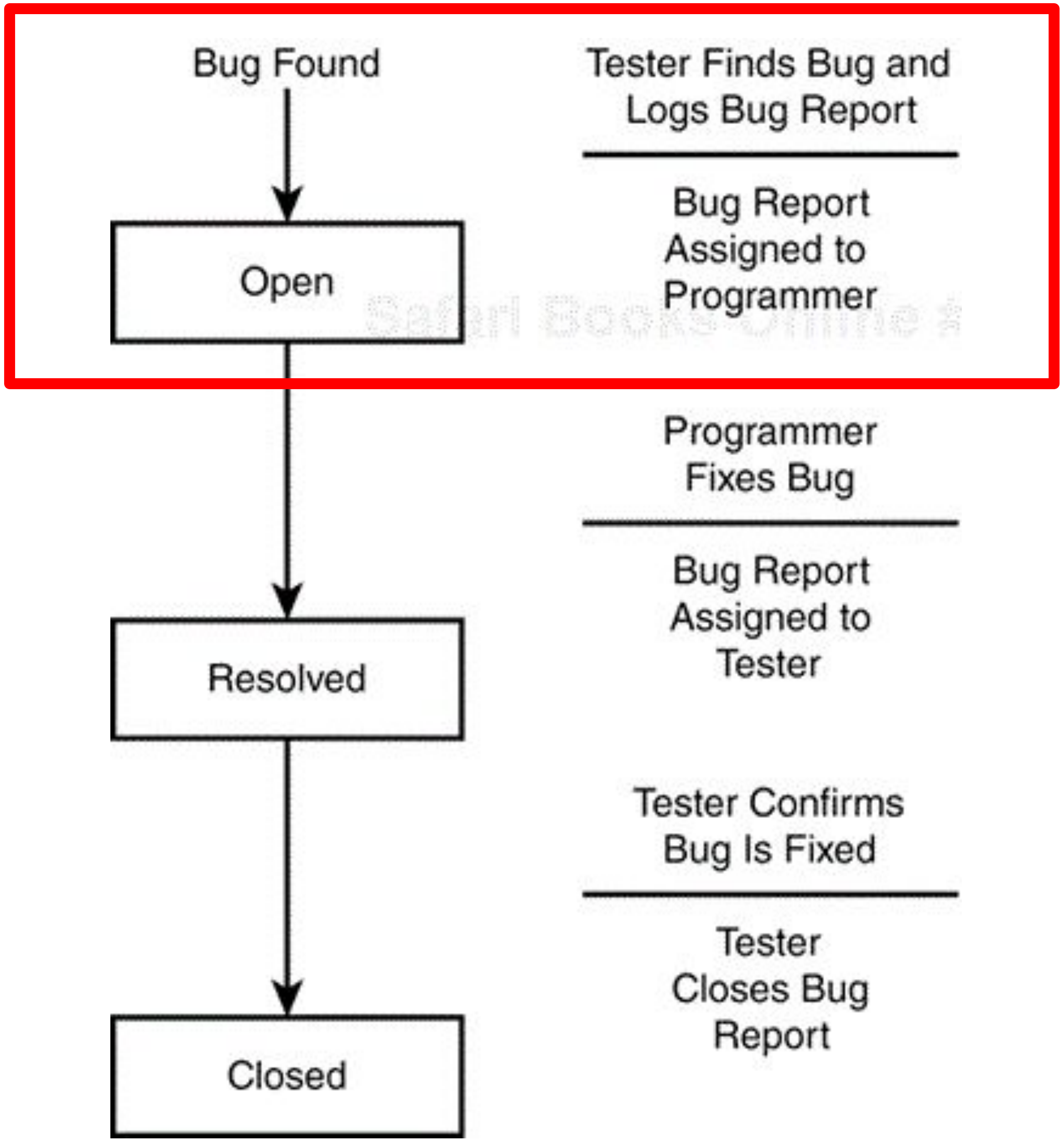
- Priority: indicates how much emphasis should be placed on fixing the bug.
- Priority possible values:
  1. Immediate fix: blocks further testing, very obvious.
  2. Must fix before the product is released.
  3. Should fix when the time permits.
  4. Would like to fix but the product can be released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

Bugs Are NOT Equal (Cont’d)

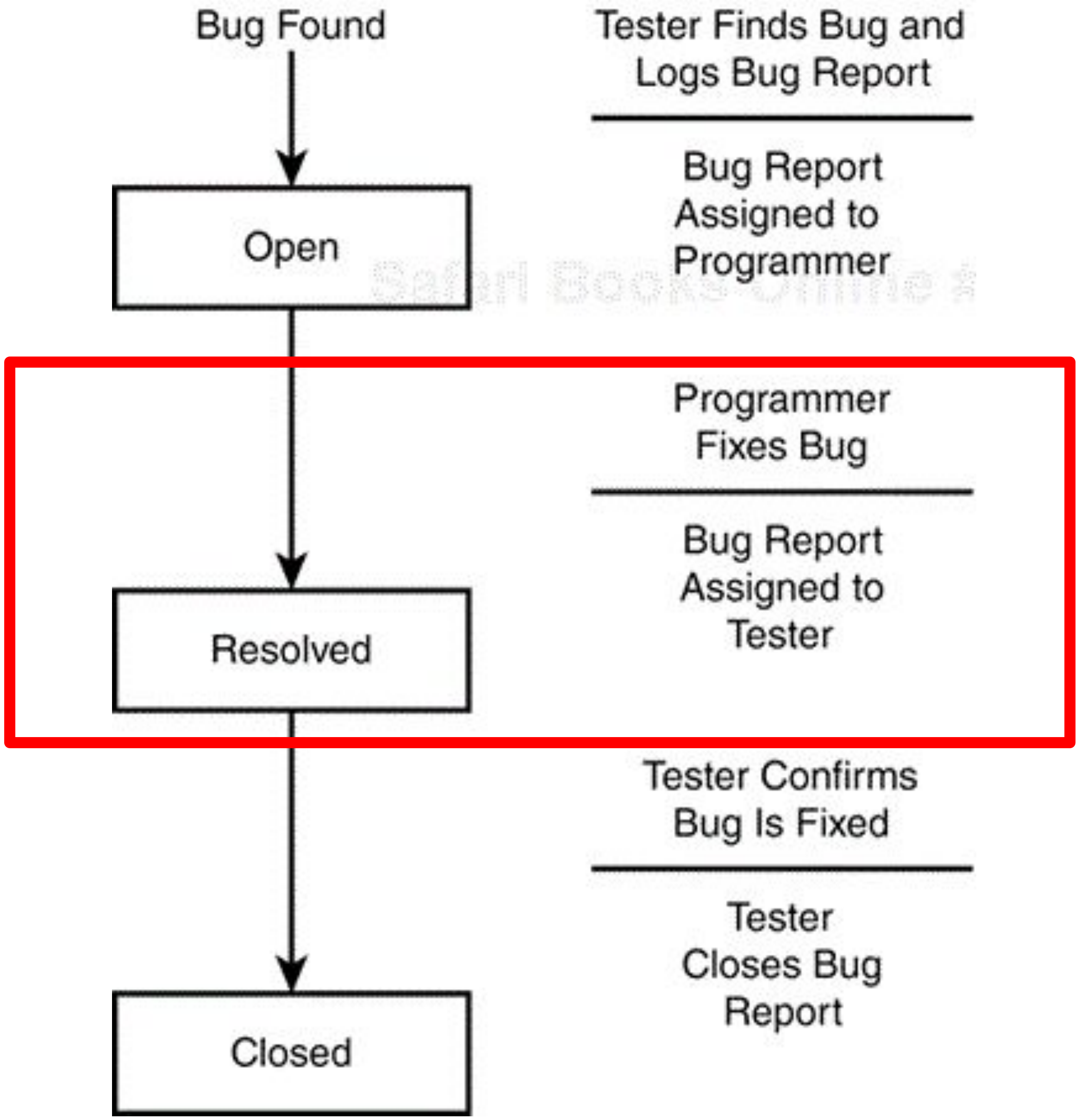
What are the severities/priorities of the bugs below?

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

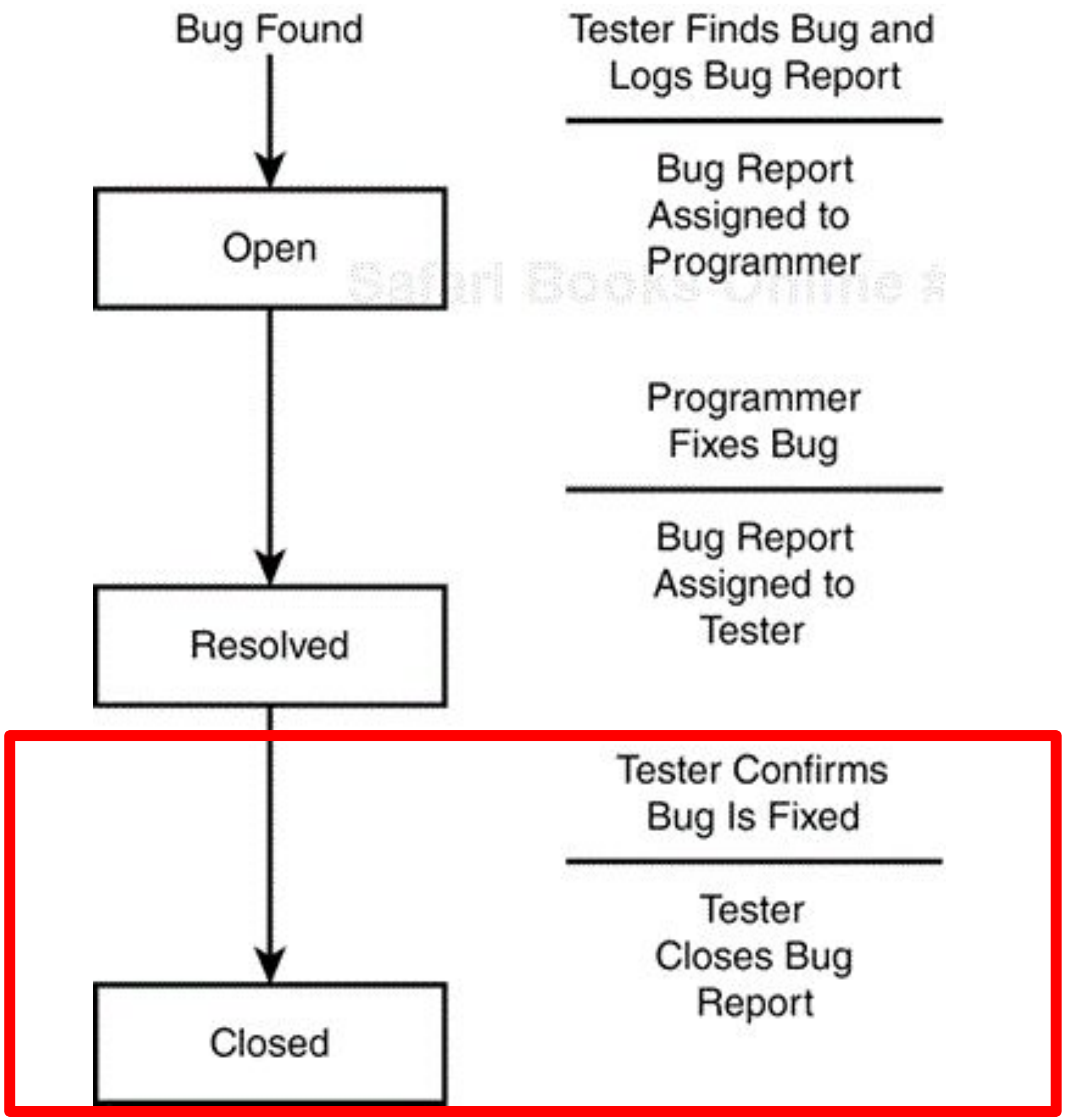
A Bug's Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Bug Report Assigned to Programmer
Open
Programmer Fixes Bug
Bug Report Assigned to Tester
Resolved
Tester Confirms Bug Is Fixed
Tester Closes Bug Report
Closed

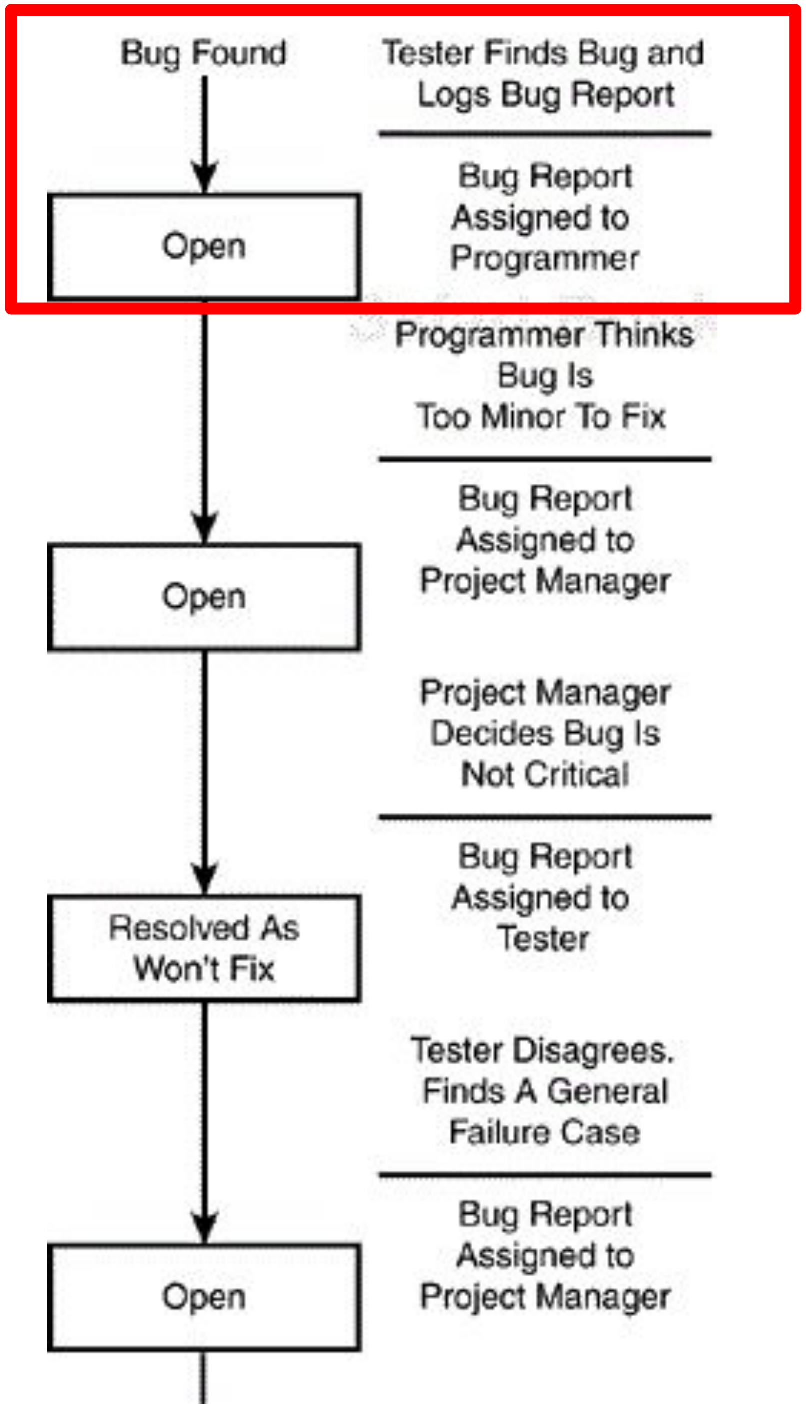
A Bug’s Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Fixes Bug
Resolved
Bug Report Assigned to Tester
Tester Confirms Bug Is Fixed
Closed
Tester Closes Bug Report

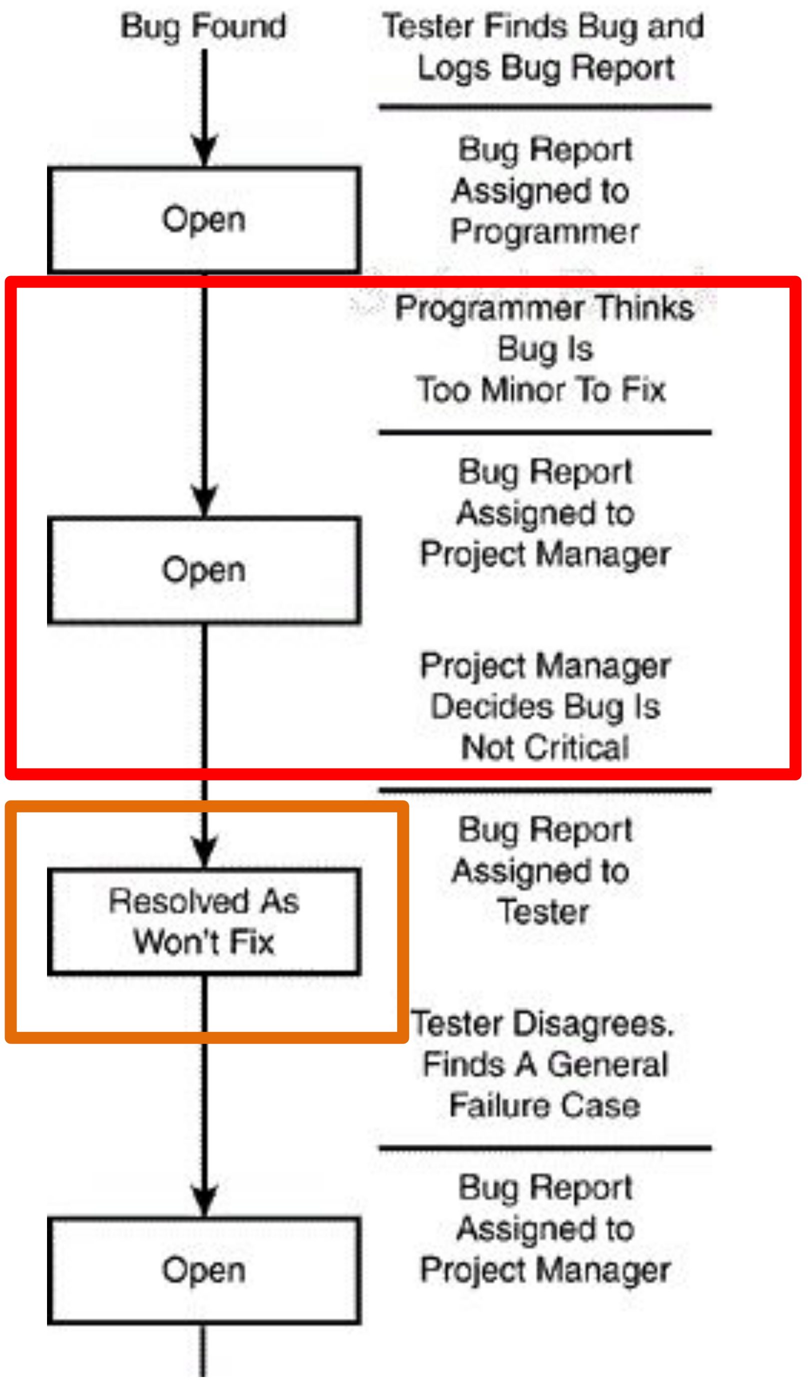
A Bug's Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Fixes Bug
Resolved
Bug Report Assigned to Tester
Tester Confirms Bug Is Fixed
Tester Closes Bug Report
Closed

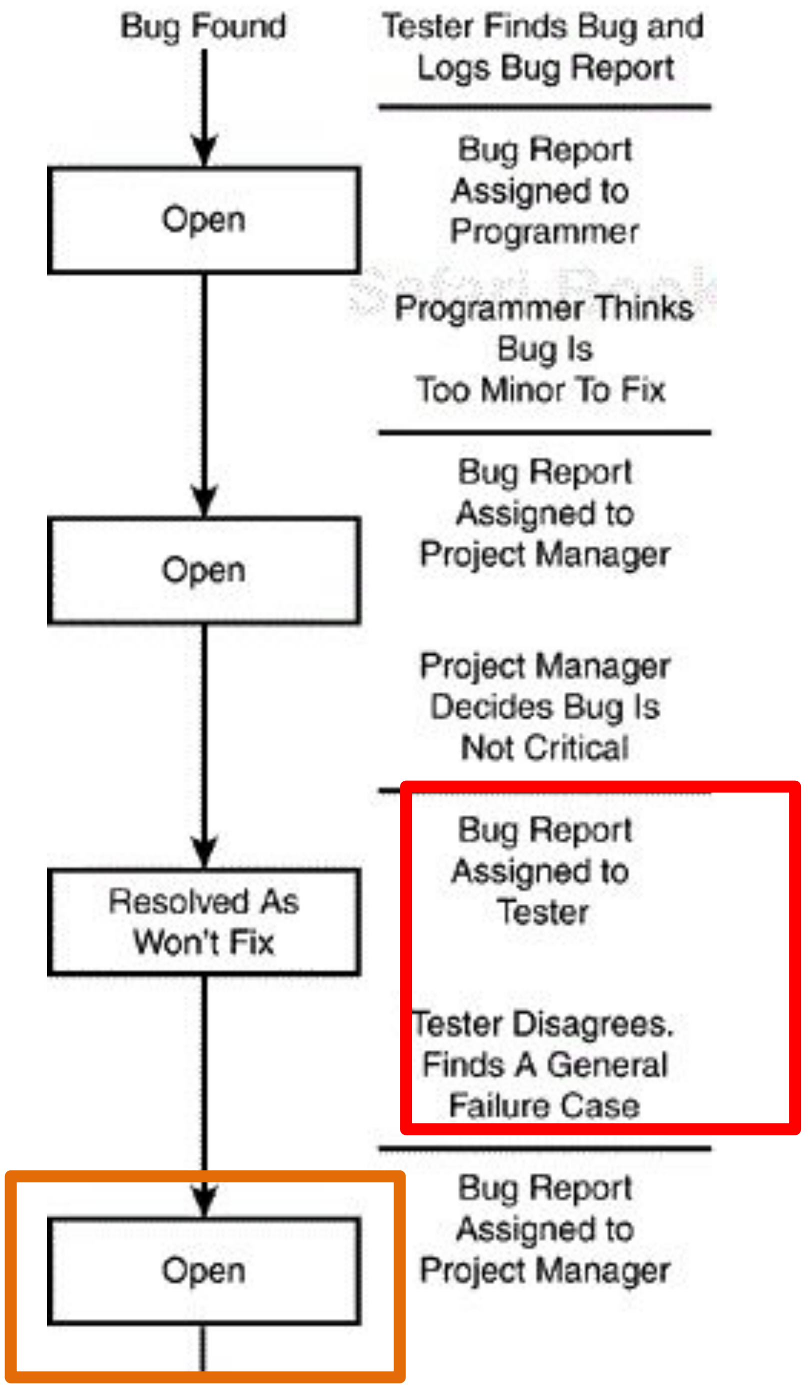
A complicated Bug's Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Thinks Bug Is Too Minor To Fix
Bug Report Assigned to Project Manager
Open
Project Manager Decides Bug Is Not Critical
Bug Report Assigned to Tester
Resolved As Won't Fix
Tester Disagrees. Finds A General Failure Case
Bug Report Assigned to Project Manager
Open

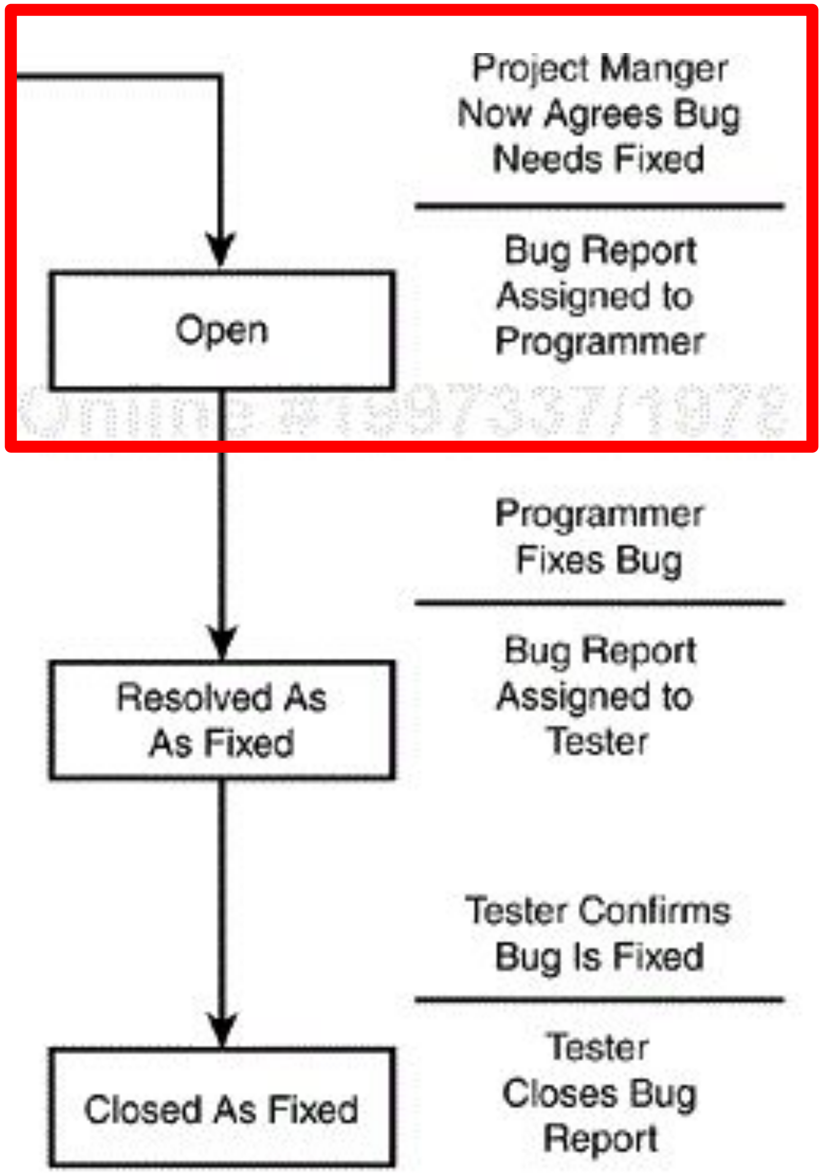
A complicated Bug's Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Thinks Bug Is Too Minor To Fix
Open
Bug Report Assigned to Project Manager
Project Manager Decides Bug Is Not Critical
Resolved As Won't Fix
Bug Report Assigned to Tester
Tester Disagrees. Finds A General Failure Case
Open
Bug Report Assigned to Project Manager

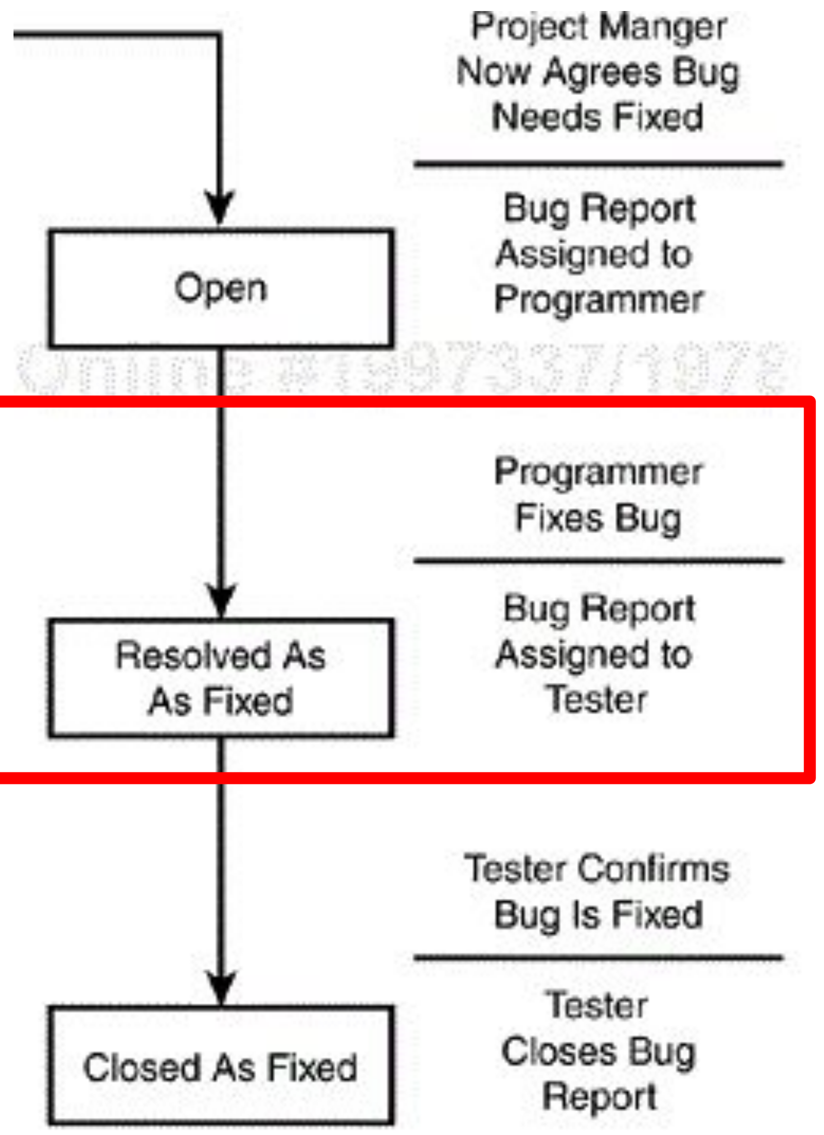
A complicated Bug's Lifecycle


Bug Found
Tester Finds Bug and Logs Bug Report
Open
Bug Report Assigned to Programmer
Programmer Thinks Bug Is Too Minor To Fix
Open
Bug Report Assigned to Project Manager
Project Manager Decides Bug Is Not Critical
Resolved As Won't Fix
Bug Report Assigned to Tester
Tester Disagrees. Finds A General Failure Case
Open
Bug Report Assigned to Project Manager

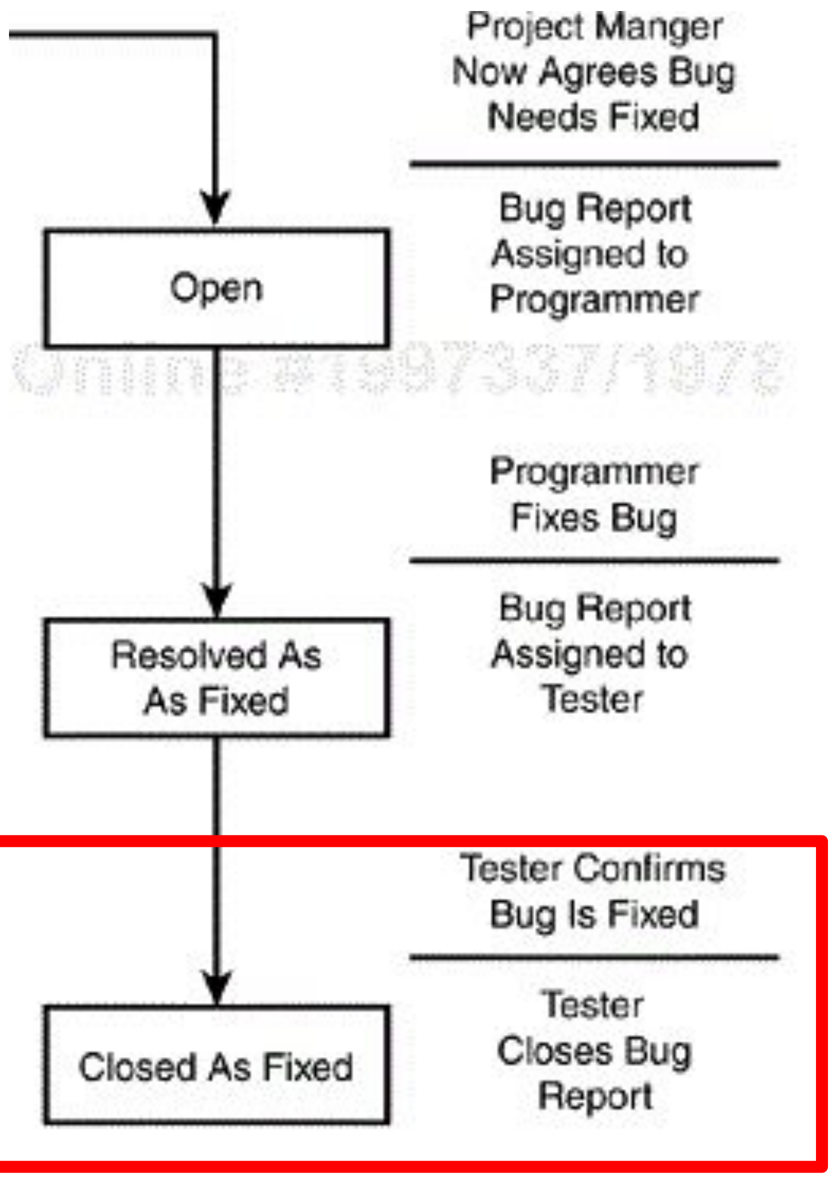
A complicated Bug's Lifecycle


Project Manger Now Agrees Bug Needs Fixed
Bug Report Assigned to Programmer
Open
Programmer Fixes Bug
Bug Report Assigned to Tester
Resolved As As Fixed
Tester Confirms Bug Is Fixed
Tester Closes Bug Report
Closed As Fixed

A complicated Bug's Lifecycle


Project Manger Now Agrees Bug Needs Fixed
Bug Report Assigned to Programmer
Open
Programmer Fixes Bug
Bug Report Assigned to Tester
Resolved As As Fixed
Tester Confirms Bug Is Fixed
Tester Closes Bug Report
Closed As Fixed

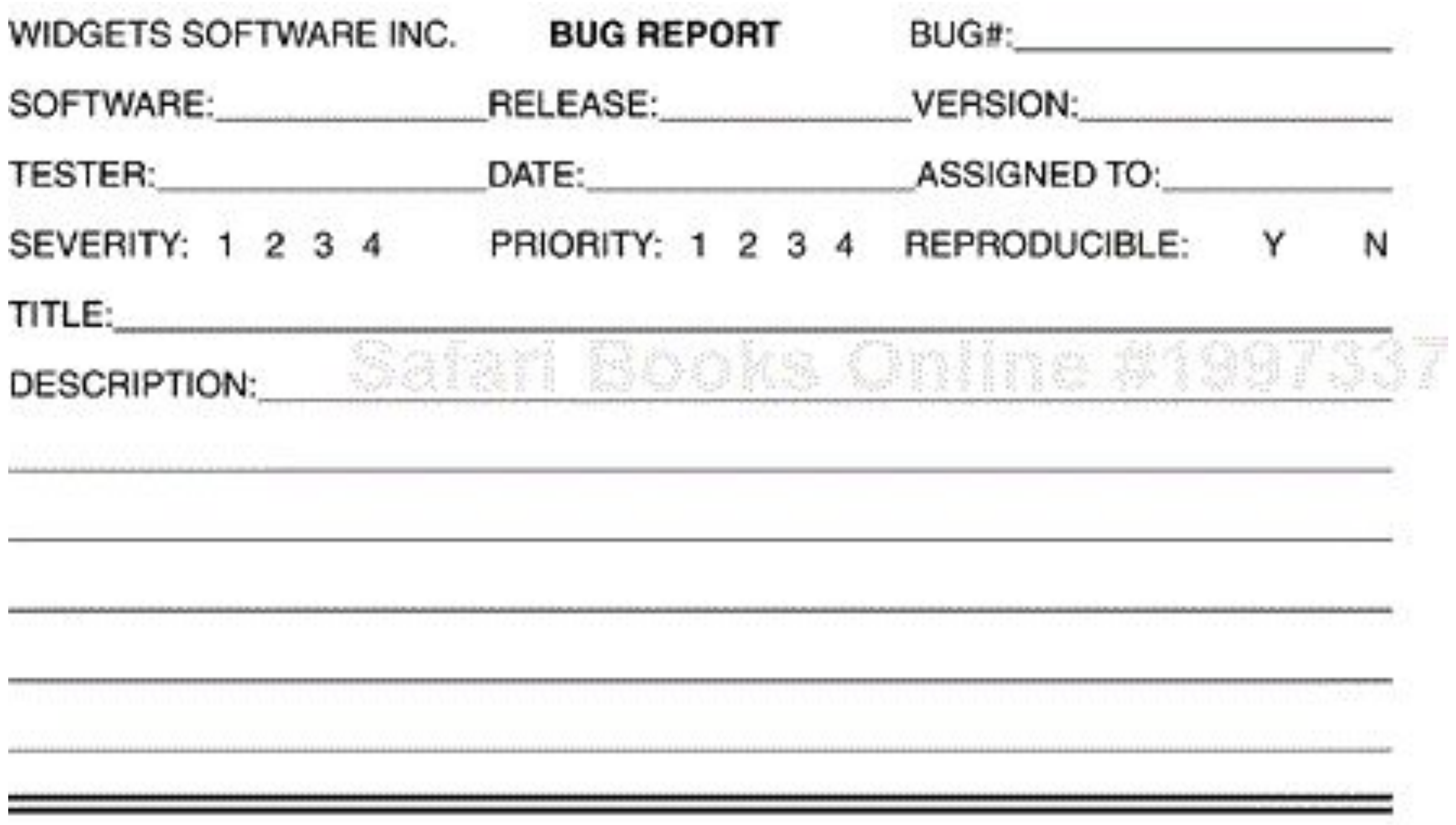
A complicated Bug's Lifecycle



Project Manger Now Agrees Bug Needs Fixed
Bug Report Assigned to Programmer
Open
Programmer Fixes Bug
Bug Report Assigned to Tester
Resolved As As Fixed
Tester Confirms Bug Is Fixed
Tester Closes Bug Report
Closed As Fixed

Bug/Defect Tracking Systems

- A bug tracking system keeps track of reported bugs.



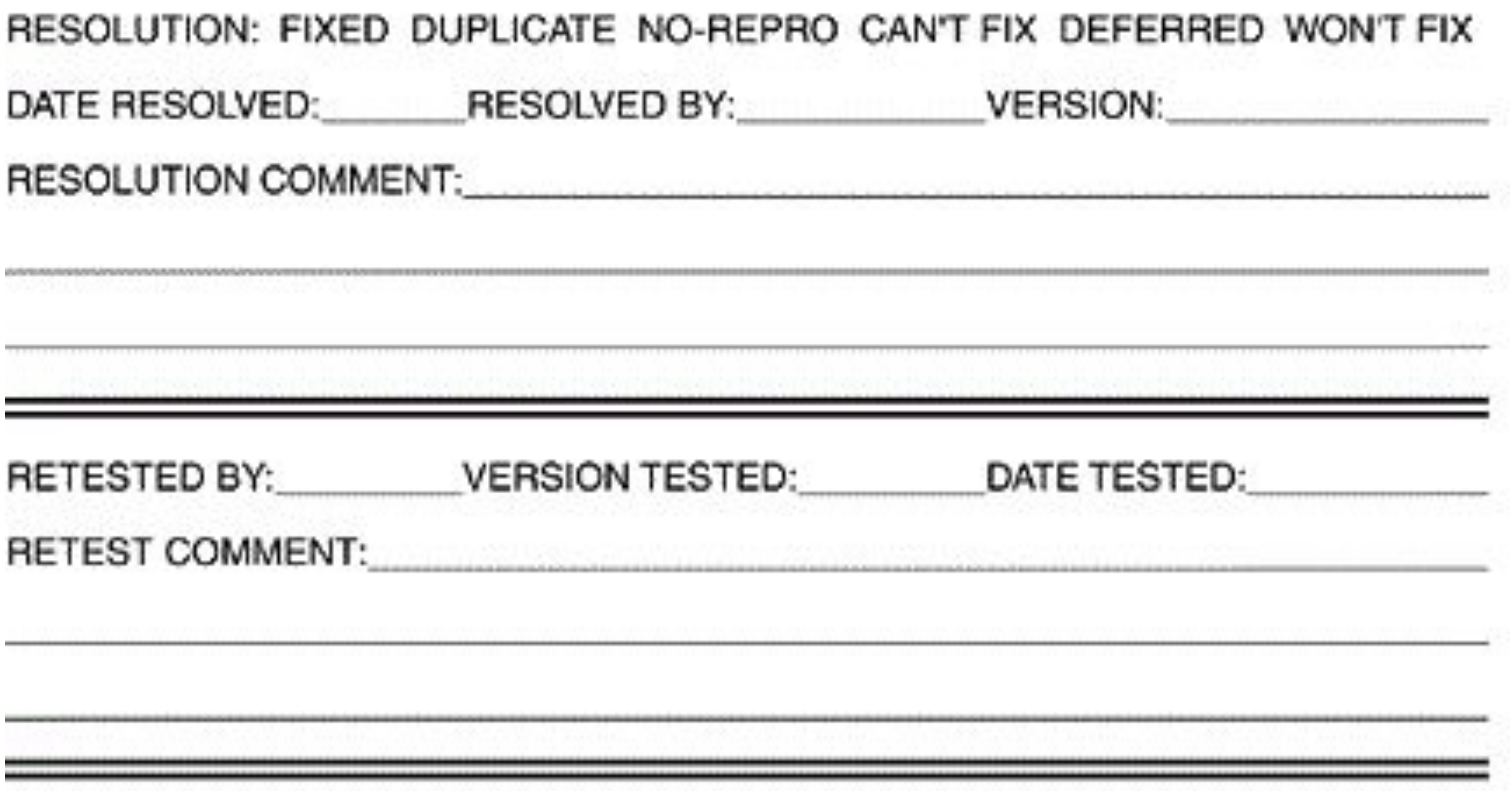
WIDGETS SOFTWARE INC.	BUG REPORT	BUG#:________
SOFTWARE:________	RELEASE:________	VERSION:________
TESTER:________	DATE:________	ASSIGNED TO:________
SEVERITY: 1 2 3 4	PRIORITY: 1 2 3 4	REPRODUCIBLE: Y N

TITLE:________________

DESCRIPTION:________________

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
  - Githib 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=474525

Required Readings

- Ron Patton. Software Testing, 2nd 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.