Issue tracking

EECS 2311 - Software Development Project

Click to edit Master text styles
Second level
Third level
Fifth level

Wednesday, March 11, 2020

YORK UNIVERSITÉ UNIVERSITY

Issue tracking

- Once a system is released, users will find bugs or will want more features
- For large systems, managing these requests can become a time consuming task
- An issue tracking system can help. There are lots of them out there.
- One of them is Bugzilla…

The following slides are based on a slide set by Mikko Rusama

What is Bugzilla?

- Bugzilla is a bug- or issue-tracking system.
  - Bug-tracking systems allow individual or groups of developers effectively to keep track of outstanding problems with their product.
- An Open Source product
  - covered by the Mozilla Public License
- Written in Perl, uses MySQL database
- See: http://www.bugzilla.org

Who is using Bugzilla?

- Mozilla
- Linux Kernel
- Apache
- Open Office
- Eclipse
- NASA
- Red Hat Software
- More than 1000 other companies

Before entering a bug

- Make sure the bug has not been previously reported!
  - Use the Bugzilla Query Form
  - For more information, see link on course webpage
- Next, be sure that you've reproduced your bug using the latest build released
  - Development process may produce new builds even daily, and the bug you've found may already have been fixed.

Useful bug report qualities

- Reproducible
  - If a developer can't see it or conclusively prove that it exists, they will probably stamp it "WORKSFORME" or "INVALID", and move on to the next bug. Every relevant detail you can provide helps.
- Specific
  - The quicker the developer can isolate the issue to a specific problem, the more likely it'll be expediently fixed.
  - If a programmer or tester has to decipher a bug, they may spend more time cursing the submitter than solving the problem

A useful bug report

- Useful bug reports are ones that get bugs fixed!
- Be non-judgmental in reporting bugs.
  - Bug reports need to be non-judgmental, non-personal and non-inflammatory.
  - Reports should be written against the product, not the person, and state only the facts.
- Let's look at Eclipse Bugzilla…

Where did you find the bug?

- **Product** - In which product did you find the bug?
- **Version** - In which product version did you find the bug?
- **Component** - In which component does the bug exist?
- **Platform** - On which hardware platform did you find this bug?
- **OS** - On which Operating System (OS) did you find this bug?

Severity

- How damaging is the bug?
- Each software project can specify the severities it plans to use

Severity Values

- **Blocker** - Blocks development and/or testing work
- **Critical** - crashes, loss of data, severe memory leak
- **Major** - major loss of function
- **Normal** – default value
- **Minor** - minor loss of function, or other problem where easy workaround is present
- **Trivial** - cosmetic problem like misspelled words or misaligned text
- **Enhancement** - Request for enhancement, ideas

Priority

- This field describes the importance and order in which a bug should be fixed.
- Utilized by the managers and programmers or engineers to prioritize their work to be done.
- The available priorities are:
  - P1 Most important
  - P2
  - P3
  - P4
  - P5 Least important

Following up on the bug

- Assigned To - Which engineer should be responsible for fixing this bug?
- Cc - Who else should receive e-mail updates on changes to this bug?
- You would not normally change either of these fields from their default values!

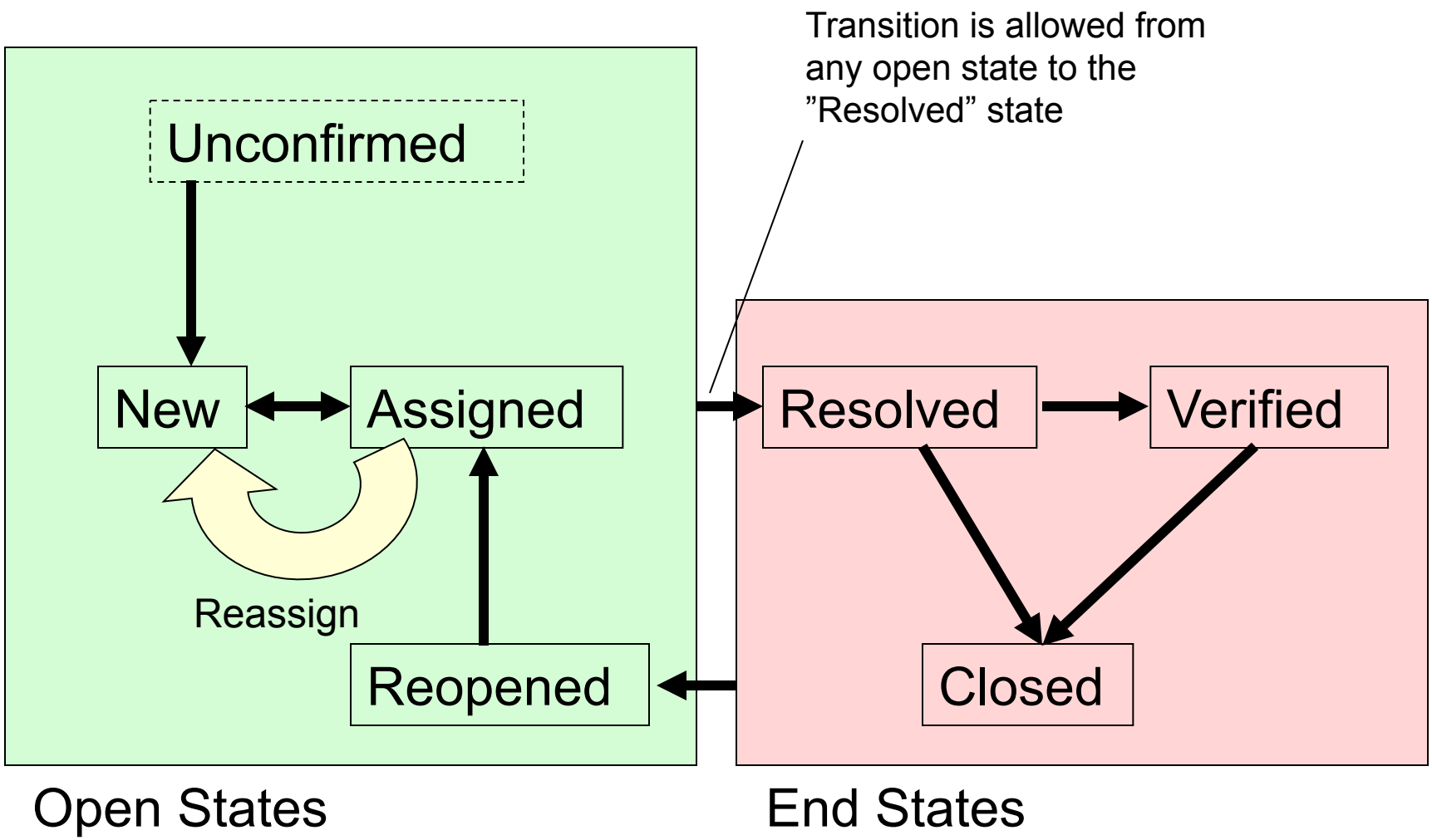
What can you tell about the bug?

- **Summary** - How would you describe the bug, in approximately 60 or fewer characters?
- **Description** - What else can you tell the engineer about this bug?

Description must include:

- **Steps to Reproduce** - The minimal set of steps necessary to trigger the bug. Include any special setup steps.
- **Actual Results** - What the application did after performing the above steps.
- **Expected Results** - What the application should have done, were the bug not present.
- **Build Date & Platform** - Date and platform of the build that you first encountered the bug in.
- **Additional Builds and Platforms** - Whether or not the bug takes place on other platforms or browsers.

Bug Life-Cycle



Bug Status – Open States

- **NEW** - This bug has recently been added to the assignee's list of bugs and must be processed.
  - Bugs in this state may be accepted, and become ASSIGNED, passed on to someone else, and remain NEW, or resolved and marked RESOLVED.
- **ASSIGNED** - This bug is not yet resolved, but is assigned to someone who thinks they can fix it.
  - From here bugs can be given to another person and become NEW, or resolved and become RESOLVED.

Bug Status – Open States

- **REOPENED** - The bug was once resolved, but the resolution was deemed incorrect.
  - For example, a WORKSFORME bug is REOPENED when more information shows up and the bug is now reproducible. From here bugs are either marked ASSIGNED or RESOLVED.

Bug Status – Unconfirmed State

- **UNCONFIRMED** - Nobody has validated that this bug needs to be fixed.
  - Users who have the correct permissions may confirm this bug, changing its state to NEW.
  - A bug may be directly resolved and marked RESOLVED but usually a bug will be confirmed by the person to whom it is assigned.
  - Usually, an UNCONFIRMED bug will be left unconfirmed until someone has verified that the bug the reporter submitted actually occurs.
- Bugzilla administrator may specify the number of votes a bug in this product needs to automatically get out of the UNCONFIRMED state.

Bug Status – End States

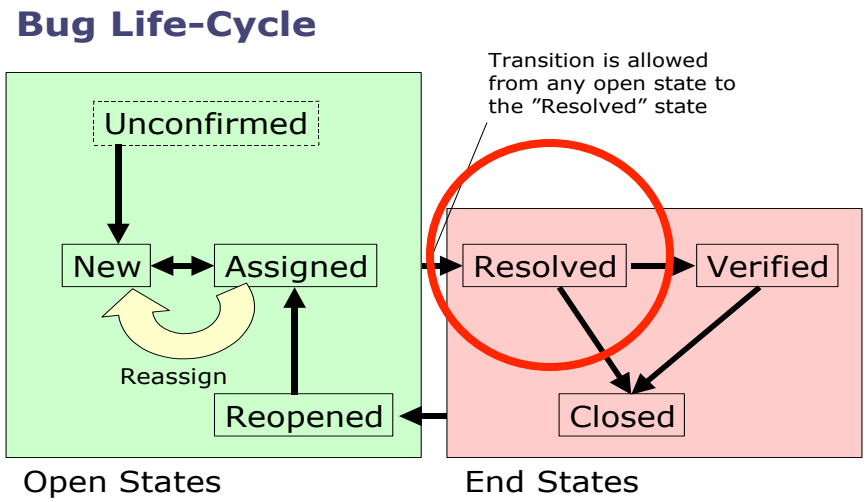
- **RESOLVED** - A resolution has been made, and it is awaiting verification by the QA.
  - From here bugs are either re-opened and become REOPENED, are marked VERIFIED, or are closed for good and marked CLOSED.
- **VERIFIED**- QA has looked at the bug and the resolution and agrees that the appropriate action has been taken.
  - Bugs remain in this state until the product they were reported against actually ships, at which point they become CLOSED.

Bug Status – End States

- **CLOSED** - The bug is considered dead, the resolution is correct, and the product the bug has been reported against is terminated or shipped.
  - Any zombie bugs who choose to walk the earth again must do so by becoming REOPENED. This state is rarely ever used.
- NOTE: Resolution values can only be specified for bugs being in one of the end states!

Resolution

- The resolution field indicates what happened to this bug.
- Only bugs in”Resolved” state will be marked with one of the resolutions.
- All bugs which are in one of the ”Open” states have no associated resolution.



YORK UNIVERSITÉ UNIVERSITY

Resolution

- **FIXED** - A fix for this bug exists and has been tested.
- **INVALID** - The problem described is not a bug.
- **WONTFIX** - The problem described is a bug which will never be fixed.
- **DUPLICATE** - The problem is a duplicate of an existing bug. Marking a bug duplicate requires the bug number of the duplicate and that number will be placed in the bug description.
- **WORKSFORME** - All attempts at reproducing this bug were futile, reading the code produces no clues as to why this behaviour would occur.

Github issue tracker

- Much simplified issue tracker
- Can submit issues in free form
- Can tag each issue
- Can assign issues to contributors
- Can see a list of issues
  - Can filter by author, tag etc.
- Each issue page is a discussion

Lab task

- **Each student** has received two links to repositories
- **Each student** must submit 3 issues for each project (for a total of 6 issues)
- These will ideally be bug reports, but if you can't find bugs, submit feature requests
- You must provide as many of the fields that we discussed as appropriate
  - Must have: OS, release tested, Java version, description, steps to reproduce, actual vs expected results
- In the lab, show the issue pages to the TA and reproduce the bugs for them