Software Testing, Quality Assurance and Maintenance Winter 2010

Lecture 32 — March 17, 2010

Patrick Lam *version 1*

Reporting Bugs

Goal of reporting bugs:

How?[1][2]

Properties of Important Bugs

- Bug is sufficiently general to affect many users (easily reproducible).
- Bug has severe consequences (crashes, dataloss).
- Bug is new to most recent version.

Reproducibility

Developers can't fix problems they can't observe.

- Be maximally specific in describing steps to reproduce.
- Write the steps down as soon as possible, before you forget.
- Try to find a minimal testcase that demonstrates the problem.

[1] `http://blog.cleverelephant.ca/2010/03/how-to-get-your-bug-fixed.html`
[2] `http://blog.threepress.org/2009/11/17/how-to-get-your-bug-fixed/`

Anatomy of a Bug Report

Bug report formats are fairly standard. Here are some fields in a Bugzilla report.

Standard bookkeeping fields:

- Bug id: automatically generated number
- Reporter: usually an email address
- Product, Version, Component: helps direct the bug to the right developers
- Platform, OS: e.g. PC/Linux, or Mac/Mac OS X 10.5.
- Severity: based on consequences of bug; examples: blocker, critical, normal, trivial, enhancement
- Assign to: person responsible for bug
- CC: list of people who track bug changes
- Keywords: e.g. crash, intl, patch, security
- Depends on/blocks: relationships between bugs
- URL: (especially applicable for browsers)
- Attachments: e.g. test cases/input files which exhibit the bug

Summary. Perhaps the most critical field: a one-line recap of the bug. Enables searching for and judgement of the bug.

Examples (some good, some bad):

- "RPM 4 installer crashes if launched on Red Hat 6.2 (RPM 3) system"
- "Back button does not work"
- "When memory cache disabled, no-store pages not displayed at all"
- "History and bookmarks completely inoperable"
- "Can't install"

Description. Should be a complete description of the bug, including:

- Overview: expand the summary, e.g. "Drag-selecting any page crashes Mac builds in NSGetFactory".

- Steps to Reproduce (key!): minimized easy-to-follow steps to trigger the bug; 1) try to reproduce the bug based on the steps you report; 2) try to describe the steps to that anyone (like the developer) can reproduce the bug.

  Consider being specific, e.g. instead of "save the document", write "File ¿ Save, select foo from dropdown, . . . ".

- Actual Results: what you see when you perform the steps to reproduce. e.g. "Application crashes. Stack trace included."
- Expected Results: what you think is correct
- Build Date and Platform: on development software, helps find the bug; include additional builds and platforms the bug might apply to.

Lifecycle-related fields. Some fields summarize the current state of the bug.

- Comments: either by the assigned developer, original reporter, or interested bystanders. Often people give additional information ("also happens on latest build") or help diagnose the problem.
- Status: e.g. UNCONFIRMED, NEW, REOPENED, VERIFIED
- Priority: for internal use by development team.

Properties of Good Bug Reports

- Reported in the database.
- Simple: one bug per report.
- Understandable, minimal, and generalizable.
- Reproducible.
- Non-judgemental. "The developers are all morons".
- Not a duplicate.

Bug Triage

Some of the fields we've seen help in identifying the most important bugs (which ones?). Consider this approach for assigning a single number to a bug which summarizes its importants ("user pain"):

`http://lostgarden.com/2008/05/improving-bug-triage-with-user-pain.html`

Reference

Cem Kaner. *Bug Advocacy.*
Retrieved from: `http://www.testingeducation.org/BBST/BBSTbugAdvocacy.htm`