IS2545 Lecture 5

Defects

Bill Laboon/Dustin Iser

Text book definition

Bug, n.: An unwanted and unintended property of a program or piece of hardware, especially one that causes it to malfunction. Antonym of feature.

-Eric S. Raymond, The Jargon File

Better definition

Some condition in a system which does one of the following:

1. Violates a requirement
2. Violates end-user expectations
3. Causes the program to malfunction or end prematurely
4. Produces an incorrect result

Where do defects come from?

- Gaps or mistakes in code
- Gaps or ambiguity in requirements
- Other:
  - Compiler broken
  - Bad hardware
  - Broken operating system
  - Gamma rays from space

A defect is visible to the user

```
// Program shall always print out “wombat”
// Program shall never print out “cephalopod”
// Is there a defect in this code?
int k = 4;
if (k > 100) {
   System.out.println(“cephalopod”);
} else {
   System.out.println(“wombat”);
}
```

Bad code != defect

While bad code should not be encouraged, it is not technically a defect and should be avoided.

Bad code is better described as ‘technical debt’.

Remember verification and validation

Defects may be entered when the system does not meet a requirement or when the system does not meet the need of the customer.

You may need to have a discussion with systems engineers or requirements analysts if the behavior of the system as defined by the requirements is not what the user expects.

Defects vs. enhancements

If the software does not meet the requirements or is unstable, then there's a defect.

If the user wants to add or modify a requirement/feature/etc., that's an enhancement request.

May be tracked similarly.

There's no such thing as a defect too small

Software is released every day with known and unknown defects.

A known bug is much better than an unknown bug.

- Images are sized 1 pixel too small.
- Delays are 1ns longer than required.
- Upon shutdown, typo in final statement.
- Seldom-used feature does not work correctly.
- Background color is slightly off.
- There should be three periods in an ellipsis, not two.

When you assume...

As a tester, ambiguity is your enemy.

The right way to report a defect.

There's no right way to report a defect.

The tester should provide enough information in a defect report for the development team to prioritize and fix the defect.

There are some things you should include:

- Summary
- Description
- Reproduction steps
- Expected behavior
- Observed behavior
- Impact
- Severity
- Priority
- Notes

Summary

A brief description of the problem

- Title does not display after clicking ‘Next’.
- CPU pegs after addition of any two cells.
- Total number of widgets in shopping cart not refreshed after removal of more that one.
- Page title is ‘Alll Entries’, should be ‘All Entries’.
- If timezone is changed during execution, idle tasks never wake up.

Description

A more detailed explanation of the problem.

- If more than one widget is removed from the shopping cart, the number of widgets is not changed from the initial value. This value is updated if the widgets are removed one at a time.

Be careful not to overgeneralize.

Reproduction steps

Specify an exact sequence of steps to reproduce the problem.

Make sure you give:

- Exact values
- Exact steps
- Exact manner of execution

It’s better to err on the side over over-specificity.

Reproduction steps

BAD: Put some things in the shopping cart. Take a couple things out.

GOOD:

1. Add three widgets to shopping cart.
2. Note number of widgets listed is 3.
3. Remove two widgets from shopping cart.
4. Observe number of widgets listed.

Expected and observed behavior

Expected behavior

- This should note, as precisely as possible, what you expected to see according to the requirements.

Observed behavior

- This should note what you actually saw.

Examples

Bad

- Expected
  - Number is correct.
- Observed
  - Number is incorrect.

Good

- Expected
  - The number of widgets in the shopping cart is 1.
- Observed
  - The number of widgets in the shopping cart is 3.

Impact

How does this defect impact the user of the software?

Bad

- The user will hate this because everything is wrong!

Good

- The user will see an incorrect number of widgets in their shopping cart, meaning they could purchase fewer widgets than they expect.

Severity

How severe is the problem?

Critical: The application is a brick.

Major: The user cannot do their job and there is no workaround.

Minor: The user is inconvenienced but there is a workaround.

Trivial: The defect is cosmetic or in a system that is not used.

Priority

How soon should the defect be addressed?

While the tester does not always set the priority, the information the tester provides will help stakeholders determine the priority.

Priority != Severity

Notes

Technical and detailed notes that can help stakeholders understand and fix the problem.

- Stacktraces
- Log file excerpts
- Environment
- Screenshots
- Anything that may be helpful to a developer while fixing the defect.

Tracking, triaging, and prioritizing defects

Once you find defects, you need to report them and eventually they need to be fixed.

Defects are usually numbered, not named.

They should have the following information:

- Identifier.
- Source - associated test case, if applicable.
- Version of software found.
- Version of software fixed, if applicable.

Lifecycle of a defect

- Discovery
- Reporting
- Triage
- Fixing
- Verification

Fixing a defect

A developer or developers will work on a fix for the bug. This may be an iterative process, with the developer and tester working hand in hand to ensure that the fix is correct and complete.

Verification

Finally, the tester verifies that the bug was actually fixed.

The tester must also validate that the fix did not break anything else.

This is called regression testing and is often the most time consuming part of the testers job.

Testers should strive to automate as much of their regression testing as possible. More on this later on in the semester.

Bug tracking tools

There are dozens of bug tracking tools and there are more being developed everyday.

- TFS
- Bugzilla
- GitHub
- Jira