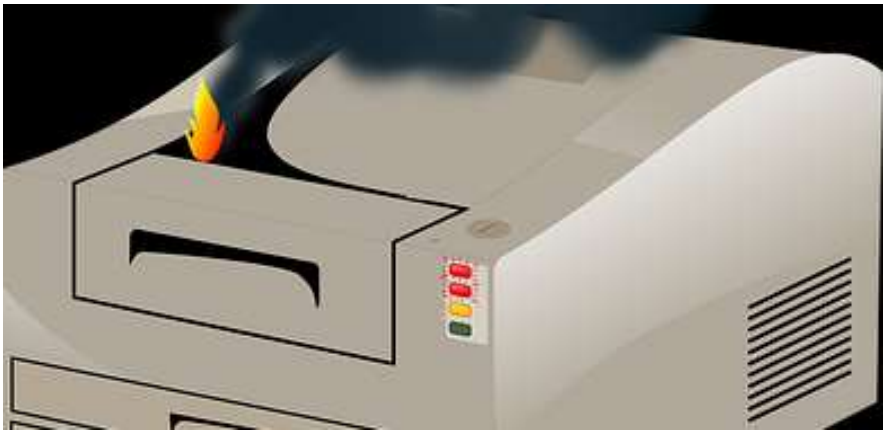
Defects



In this video, we will discuss defects. Where do they come from? What do we do with them? How do we prevent them?

Objectives

- Upon completion of this video, you should be able to:
  - Define the terms defect, error, fault, and failure
  - Describe ways of finding defects
  - Classify defects into categories
  - Prevent defects from entering your work products
  - Explain what happens when we remove defects

Here's what you should be able to do once you've finished this video.

You should be able to define the terms defect, error, fault, and failure.

You should be able to describe ways of finding defects.

You should be able to classify defects into categories.

You should be able to prevent defects from entering your work products.

You should be able to explain what happens when we remove defects

Some Definitions

- Error: A conceptual, syntactic or clerical discrepancy which results in one or more faults in the software.
- Fault: A specific manifestation of an error. A discrepancy in the software which can impair its ability to function as intended. An error may be the cause for several faults.

Glossary of Software Engineering Terminology, IEEE Std 610.12-1990

https://softwaretestingfundamentals.com/error-defect-failure/

Let's first agree on the definitions of several terms.

According to the Gossary of Software Engineering Terminology, the difference between an error and a fault is the fault is the bug in the code and the error is the bug in the process that generated the code.

For example, if a programmer misunderstood how to use a certain feature of the programming language or IDE tool, that one misunderstanding is an error that could possibly manifest itself as multiple faults in the software. These faults would need to be individually detected and repaired.

More Definitions

- Failure: A software failure occurs when a fault in the computer program is evoked by some input data, resulting in the computer program not correctly computing the required function in an exact manner

2Lloyd, D.K., and M. Lipow, Reliability, Management, Methods and Mathematics, 2nd Edition, published by the authors, 1977

- Defect: Either a fault or discrepancy between code and documentation that results in mischief in the testing, installation, maintenance, or use of software

Dunn, Robert, and Ullman, Richard, Quality Assurance for Computer Software, McGraw-Hill, New York, 1982

When we run software that contains faults, sometimes it will fail. This failure may be as minor as computing the wrong result or as major as bringing down the entire system.

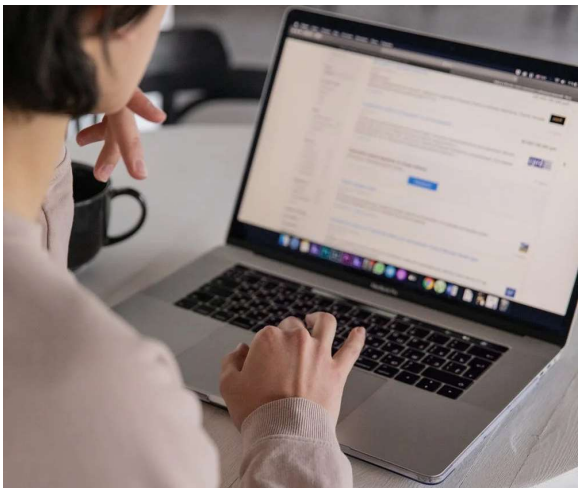
Sometimes we use the term, defect, in place of fault. The definitions are very close.

Errors Lead to Defects Lead to Failures

To summarize,, errors cause defects, which, in turn, can result in failures.

Finding Defects

- Ways to find defects:
  - Reviews
  - Walkthroughs
  - Formal Inspections
  - Testing
  - Bug reports



Because software is written by humans, there is always the chance that defects will be introduced into the code.

We have devised multiple ways to detect defects.

Reviews are performed by a group of people, including the developer of the code. There are multliple levels of formality in these reviews. An informal review is called a walkthrough. We basically walk through the design or code in the group. Usually, the developer is the one leading the walkthrough. Because there are multiple sets of eyes looking at the work product, there is a higher probability that if there is a defect, someone will spot it.

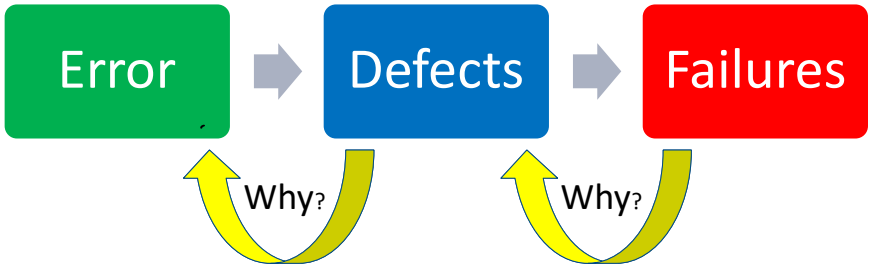
A more formal type of review is a formal inspection. These types of reviews are performed under well defined procedures and rules. The members of the inspection team have assigned roles to play. The number and types of defects found are analyzed as Quality Assurance data.

Much of the testing we do on our software has the objective of finding defects, particularly at lower levels of testing.

All of this is done internally. What we don't prefer to see is for the customer to find defects. When they submit a bug report, what they are describing is a failure. It is up

to the developers to figure out what defects led to the failure.

Finding Errors – Root Cause Analysis



What we really need to do is to figure out why the defects happened in the first place.

This is called root cause analysis. Basically, we just keep asking the question, "Why?"

Starting with a failure, we ask why did it happen? What was the defect that led to the failure?

Then we ask, why did the defect occur? What error led to the defect?

Here's a little example. Suppose an airplane is flying and loses part of its rudder. The pilot is a hero and is able to land the plane safely, What happens next?

Well, if the airplane was run by a software company, we might just duct tape the broken part back together and continue flying the plane and hope nobody notices.

What the airine does, though, is figure out what caused the part to fail, fix it, and then inspect all other aircraft of the same type. This may require grounding the planes until this inspection is done, depending on the severity of the failure. If necessary, if the same defect is found in other planes, they would be fixed as well.

But the process doesn't stop there. We would ask the question, "Why did the defect occur in the first place?" What was the error that led to the defect? Suppose the part

broke because of rusted bolts. What could cause the bolts to rust? Maybe it was due to environmental conditions. The planes weren't designed to fly in salty sea air.

Okay, why did we use bolts that could rust? What was wrong with our design process that we didn't ask this question.

For a real world example of this, look up what happened after the space shuttle Challenger blew up soon after launch.

Handling Defects

- What to do once a defect is found:
  - Record and categorize the defect
  - Identify and record the error
  - Look for similar defects and errors

We have this tendency to hide our defects and hope nobody notices them.

A better approach would be to acknowledge the defects and errors. We should look for patterns of defects and errors. If we keep making similar mistakes, we need to modify our process to avoid making these errors.

Maybe we need checklists to make sure we do all the steps in the right order Maybe we need refresher training on languages or tools. Maybe we just need to beef up our review and testing processes.

Categories of Defects

- Nature
  - functional, performance, usability, compatibility, security
- Severity
  - Critical, High, Medium, Low
- Priority
  - Urgent, High, Medium, Low

Sidorova, Tatyana. “Software Testing Basics: Types of Bugs and Why They Matter" ScienceSoft, 29 June 2020, https://www.scnsoft.com/blog/types-of-bugs

When we record our defects, it is helpful to categorize them. This will help us when we are looking for similar defects.

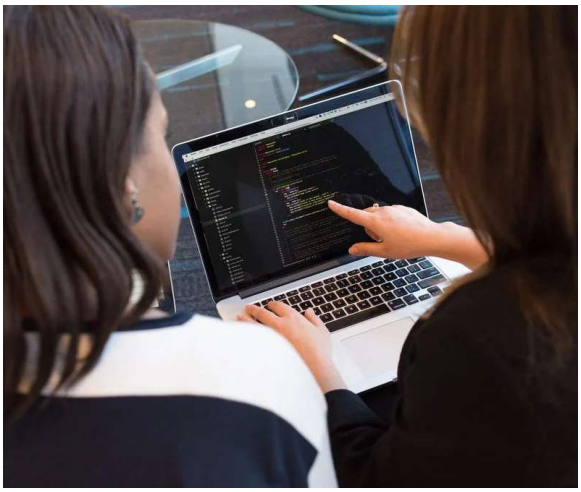
One type of categorization is by the nature of the defect. Examples are functional, performance, usability, and so forth.

We also need to identify the severity of the defect. What is the severity of the failures caused by the defects? Does it shut things down completely, or is it simply a minor annoyance, or somewhere in between?

When we assign a priority we are deciding the timing of repair of the defect. Urgent defects wil be repaired as soon as possible. The repair of low priority defects can wait until we have time.

Removing Defects

- Change control
- Regression testing
- Ripple effect



When we fix a defect, we are changing the software. These changes need to be made in a controlled manner. There needs to be a strict change control process in place. You don't want someone going in and making unapproved changes.

If you make a change to the software, you need to re-run some of the test cases to make sure you haven't accidentally broken something else. This is called regression testing.

We need to do a careful job of analyzing the changes we make to the software to avoid ripple effects. You change one thing and something else breaks. You fix that and something else doesn't work. It's like a game of whack-a-mole.

Preventing Defects

- Defect tracking and cataloging
- Continuous Process Improvement

George Santayana said, “Those who cannot remember the past are condemned to repeat it.”

To avoid making the same defects over and over again, we need to figure out the errors that lead to the defects and avoid the errors.

We have aready discussed defect tracking and cataloging and root cause analysis.

A good way to avoid errors is to modify our processes and procedures. This should be done on a continuing basis. This is referred to as continuous process improvement.

If we don’t’ continually improve our processes, we risk making the same mistakes over and over again.

There is a tremendous return on investment of doing this.

It’s a simple idea, embodied in Santayana’s aphorism as well as other wise sayings:

An ounce of prevention is worth a pound of cure.

A stitch in time saves nine.

Next

- The Quality Triangle: People, Processes, Tools

In our next video, we will examine what we call the Quality Triangle: People, Processes, and Tools