Introduction to Software Testing

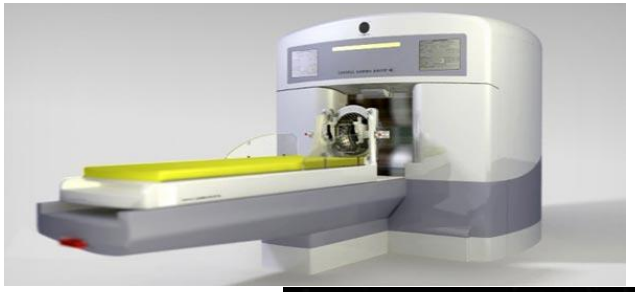
Credits: [Ammann and Offutt, "Introduction to Software Testing"]
CS 4501 / 6501 Course Materials – University of Virginia
Reading: Chapter 1 – Why do we test software?

Overview

- Software is everywhere
- Bug?
  - Fault, Error, Failure
- Software failures examples
- How important is testing?
- What is software testing?
  - Static vs. Dynamic Testing
- Validation and Verification
- Testing and Software Development Life Cycle
- Classification of software testing levels

Software is Everywhere















Bug?

'Bug' – as such little faults and difficulties are called – show themselves, and months of anxious watching, study, and labor are requisite before commercial success – or failure – is certainly reached." [Thomas Edison, 1878

[Did You Know? Edison Coined the Term "Bug", http://theinstitute.ieee.org/tech-history/technology-history/did-you-know-edison-coined-the-term-bug, IEEE 2013]

The first use of bug to generally mean a problem we were able to find is from a quote by Thomas Edison.

Bug is used informally.

Fault? Error? Or failure?

This course will try to use words that have precise, defined, and unambiguous meaning

Fault, Error, and Failure

Fault: a static defect in the software's source code

Cause of a problem

Error: An incorrect internal state that is the manifestation of some fault

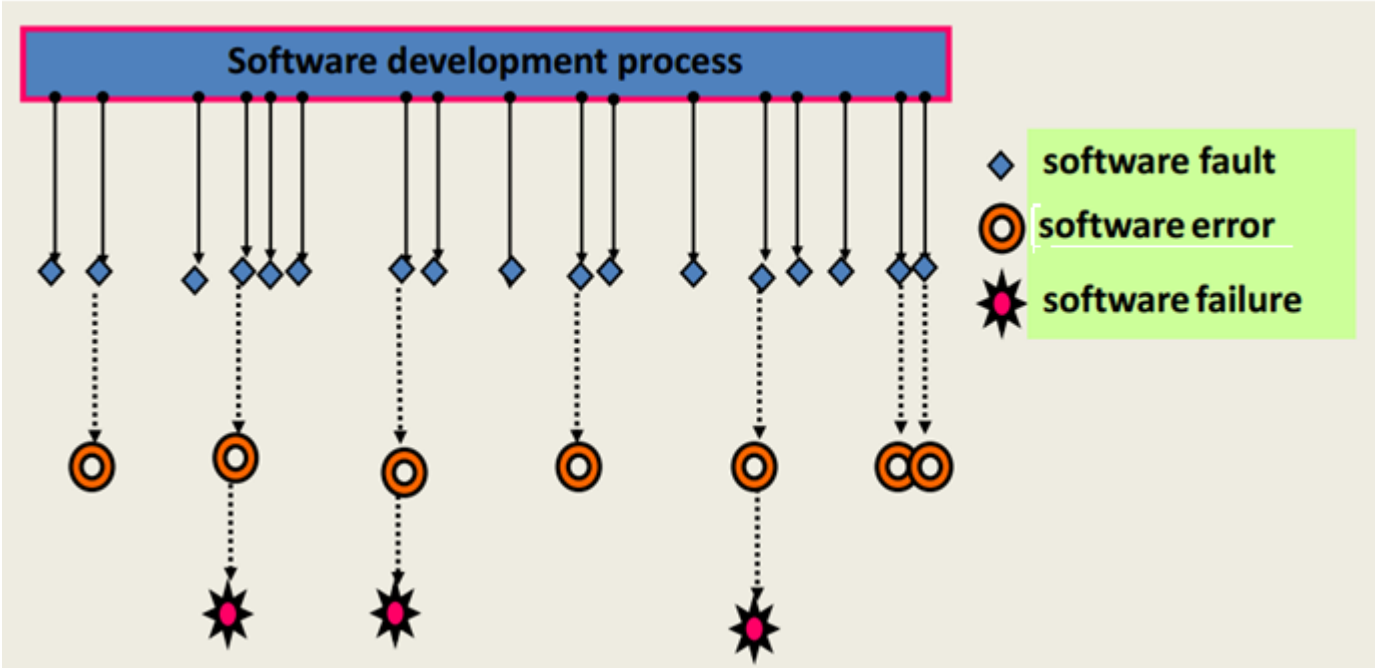
Erroneous program state caused by execution of the defect

Failure: External, incorrect behavior with respect to the requirements or other descriptions of the expected behavior

Propagation of erroneous state to the program

Note: Fault is necessary (not sufficient) condition for the occurrence of a failure

Software errors, faults and failures (Cont)



Software errors, faults and failures (Cont)

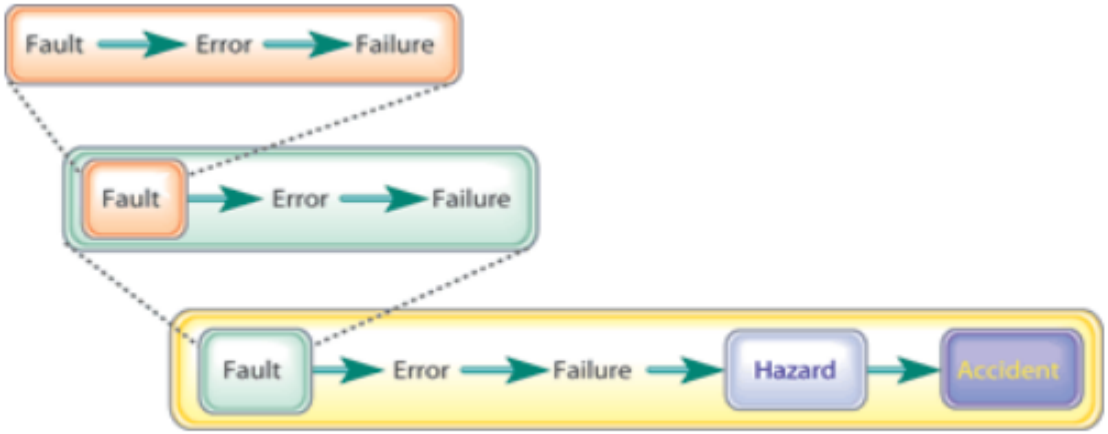


Figure 1: Fault-error-failure cascade can lead to life-threatening hazards

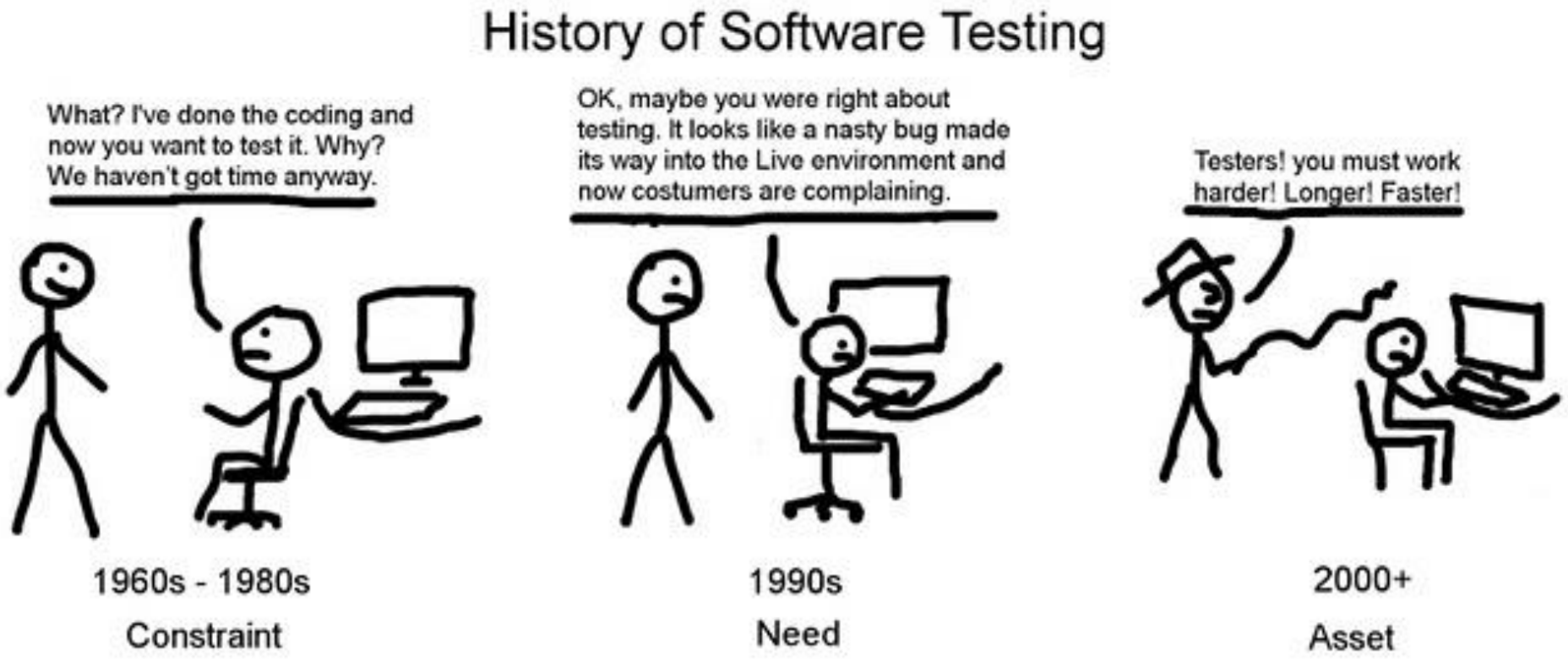
A low-level failure in some small part of a system can be viewed as a fault at another level, which can lead to errors at that level that can trigger failures that can themselves be viewed as faults at yet a higher level. If these faults are allowed to "avalanche" to system-level failures, they can lead to hazards that have the potential to threaten injury or loss of life.

Source: Architecture of safety-critical systems
https://www.embedded.com/print/4006464

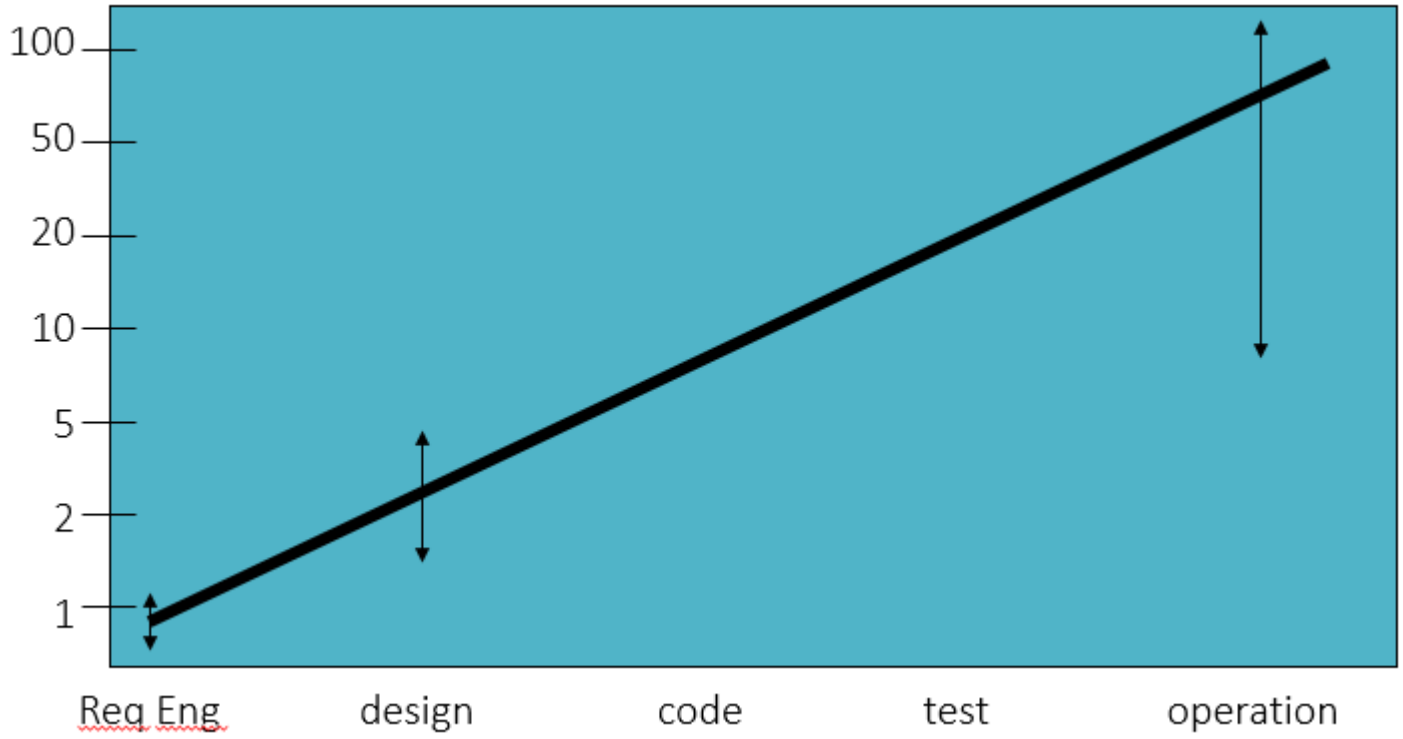
Software Failures

- 2017: 606 recorded software failures, impacting 3.7 billion people, 314 companies, $1.7 trillion in financial losses (see: https://www.tricentis.com/software-fail-watch)
- 2016: Nissan recalled 4 millions cars from the market due to software failure in the airbag sensory detectors.
- 2015: Bloomberg's trading terminal failures forced the British government to postpone $4.4 billion debt sale
- 2012: Faults in a new Knight Capital's trading software causes $440 millions
- 2003: Northeast blackout due to the alarm system in the energy management system failure, affecting 40 million people in 8 US states, 10 million people in Ontario, Canada
- 1999: NASA's Mars lander crashed due to a unit integration fault
- 1997: Ariane 5 explosion: Exception-handling bug forced self-destruct on maiden flight (64-bit to 16-bit conversion), causing $370 millions

How Important is Testing?



Relative cost of error correction



Old picture from Boehm's book.

It shows that errors discovered during operation might cost 100 times as much as errors discovered during requirements engineering. **Why so?**

Comparison of Software & Systems Cost Factors

	Software Cost Factors	Systems Cost Factors
Requirements	1X	1X
Design	5-7X	3X-8X
Build	10X-26X	7X-16X
Test	50X-177X	21X-78X
Operations	100X-1000X	29X-1615X

Table 13: Comparison of Software & Systems Cost Factors

Cost factors represent normalized costs to fix an error. These factors may be used as a "yardstick" to measure or predict the cost to fix errors in different projects.

Source: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf

Testing in the 21st Century

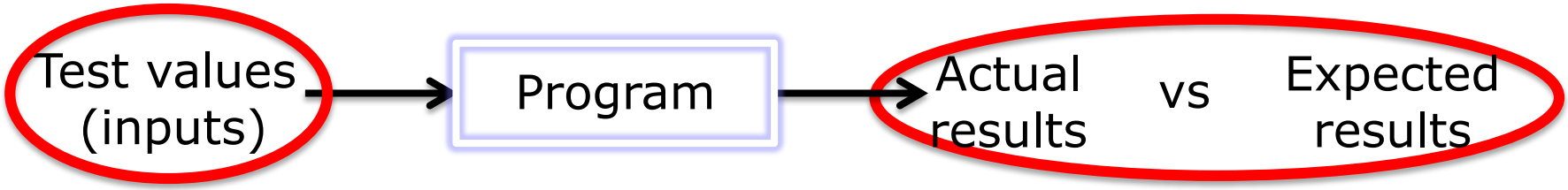
- Safety critical, real-time software (Nuclear reactor, heart lung machine)
- Embedded software (digital watch, hybrid vehicles, avionics system)
- Enterprise applications (online shopping, i.e. amazon.com)
- Security (facial recognition system, encryption)
- Web
- Software testing is getting more important

What are we trying to do when we test ?
What are our goals ?

What is Software Testing?

- Testing = process of finding input values to **check** against a software *(focus of this course)*

- Debugging = process of finding a fault given a failure

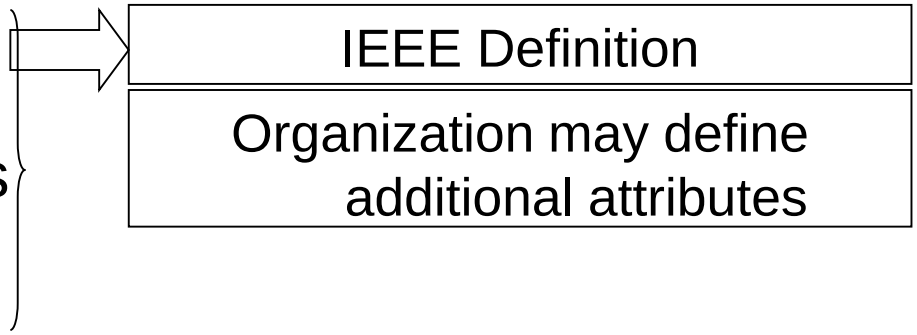


1. Testing is fundamentally about choosing finite sets of values from the input domain of the software being tested

2. Given the test inputs, compare the actual results with the expected results

Definitions

- A **TEST CASE** consists of:
  - A set of inputs
  - Execution conditions
  - Expected outputs



- A **Test** is:
  - Group of related test cases
  - Procedures needed to carry out the test case

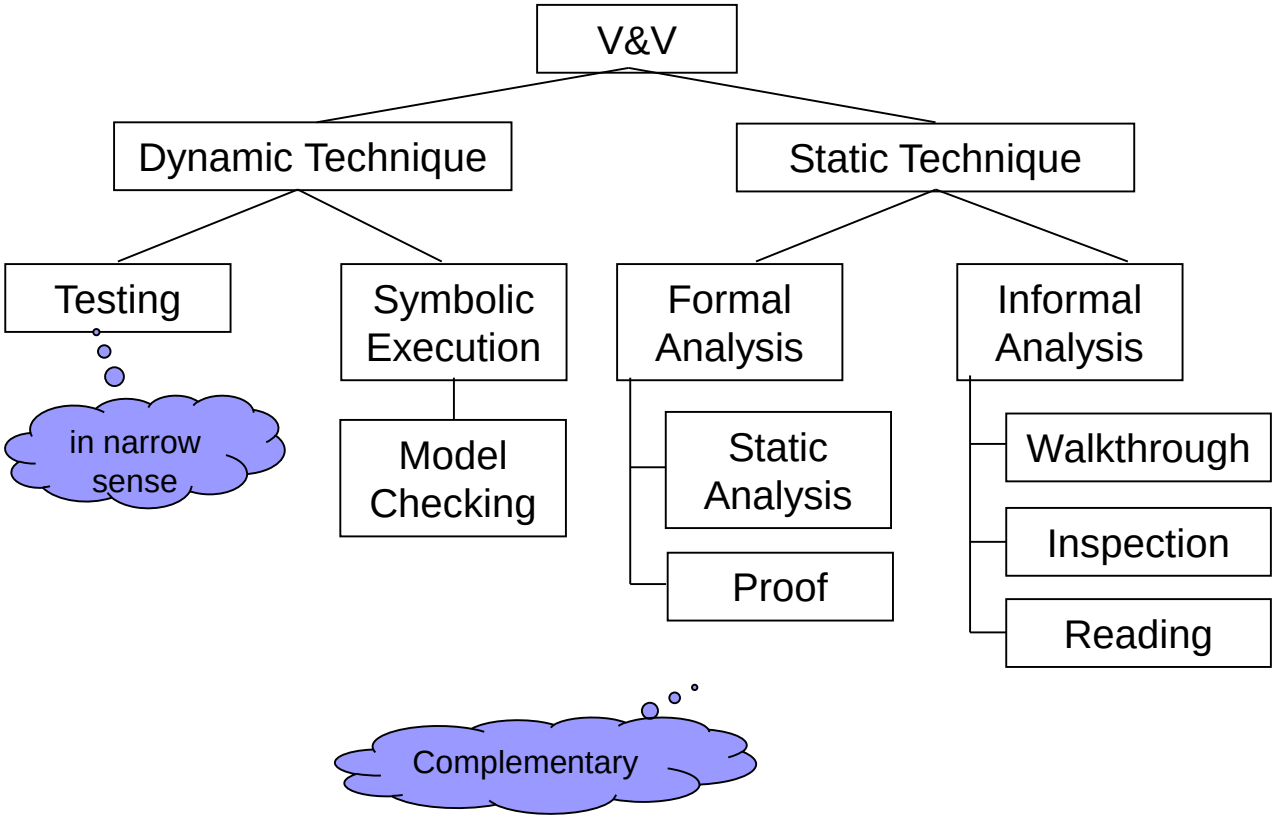
Test oracle

- An oracle is the principle or mechanism by which you recognize a problem
- Test oracle provides the expected result for a test, for example
  - Specification document
  - Formula
  - Computer program
  - Person
- In many cases it is very hard to find an oracle
  - Even the customer and end user might not be able to tell which is the correct behaviour

Static and Dynamic Testing

- **Static Testing** : Testing without executing the program
  - Software inspection and some forms of analysis
  - Effective at finding certain kinds of problems such as problems that can lead to faults when the program is modified
- **Dynamic Testing** : Testing by executing the program with real inputs

Hierarchy of V&V Techniques



Validation and Verification (IEEE)

- **Validation** : The process of evaluating software **at the end** of software development to ensure **compliance with intended usage**
  - Not done by developers, but by experts in the intended usage of the software
  - Evaluation of an object to demonstrate that it meets **the customer's expectations.** (*Did we build the right system?*)
- **Verification** : The process of determining whether the products of **a given phase** of the software development process **fulfill the requirements** established during the previous phase
  - Requires technical background on the software, normally done by developers at the various stages of development
  - Evaluation of an object to demonstrate that it meets its **specification**. (*Did we build the system right?)*

IV&V stands for "*independent verification and validation*"

IV&V (*independent verification and validation)*

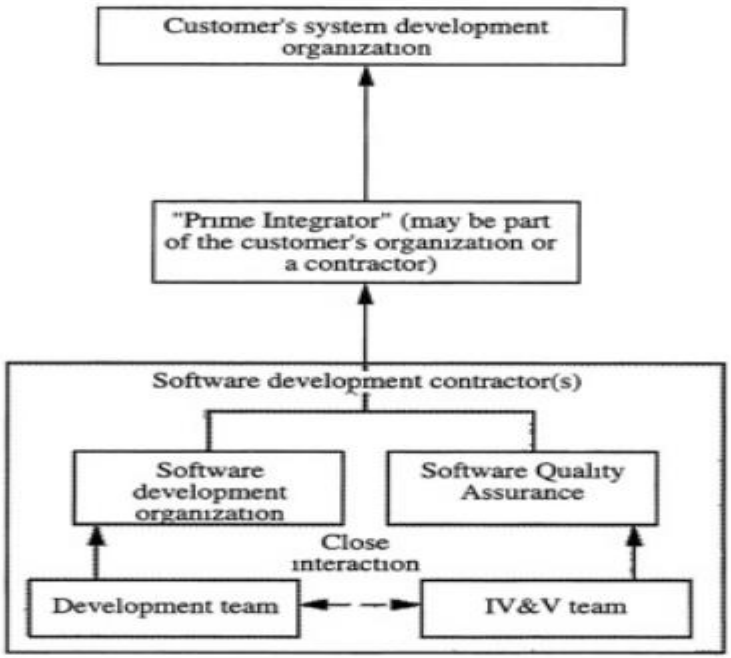
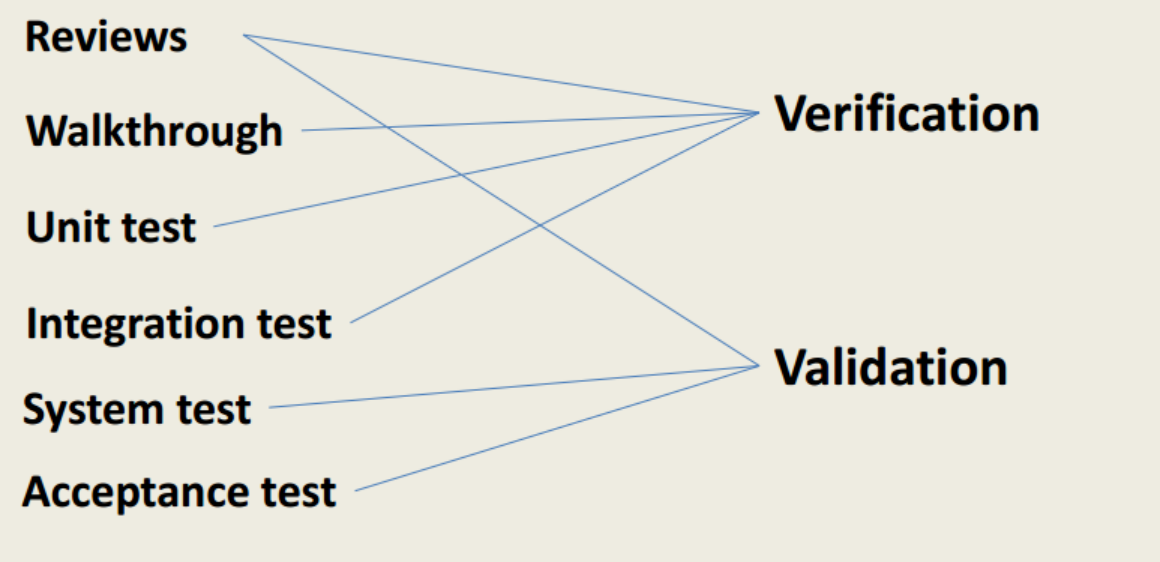


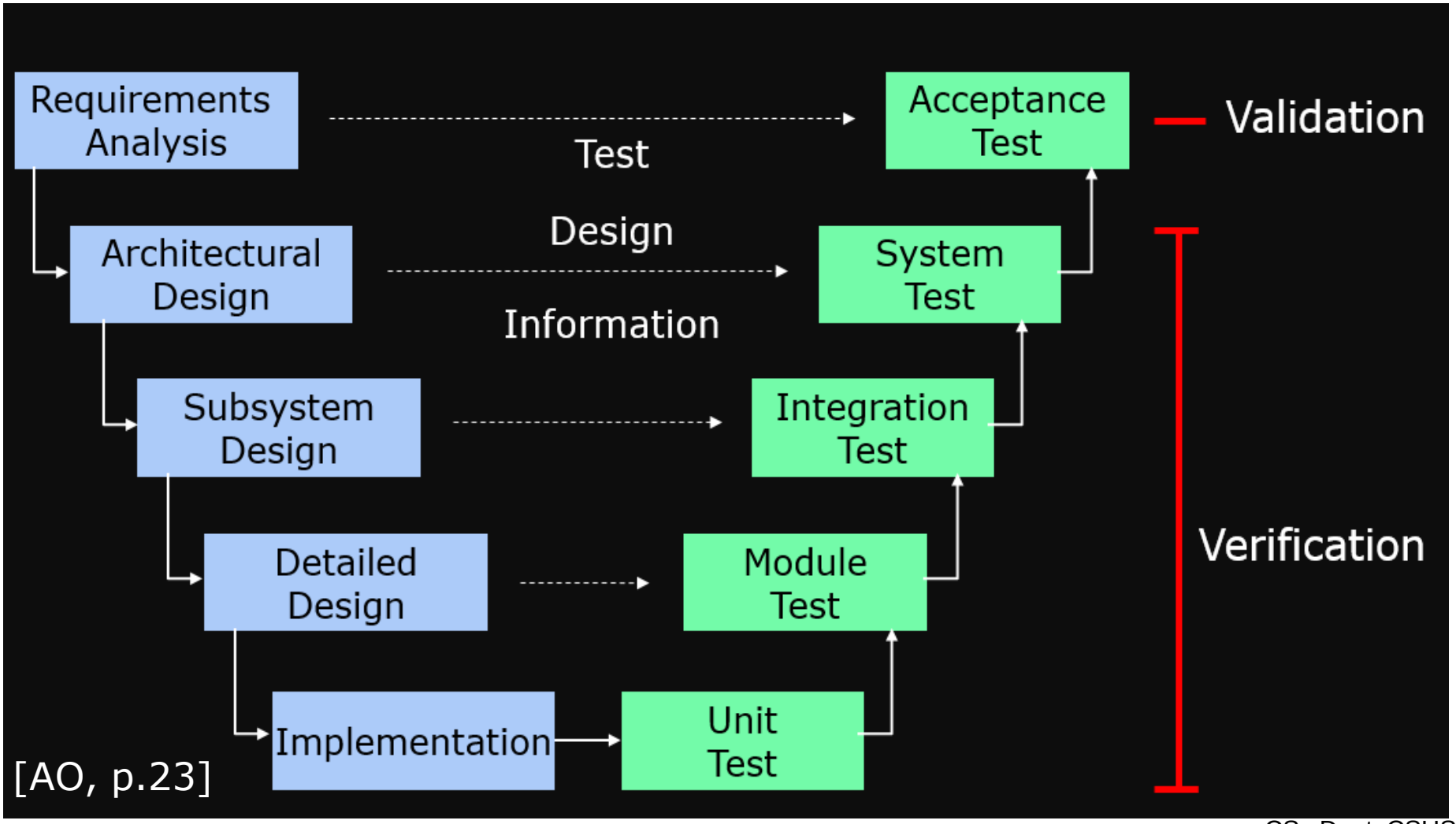
Figure 2-1c *Internal IV&V* is performed by the development contractor, but the IV&V and development teams report to different management levels within the company.

Source: National Academic Press (https://www.nap.edu/read/2222/chapter/4)

Verification vs. Validation



Testing and SW Development Lifecycle



Testing Goals Based on Test Process Maturity

Beizer's scale for test process maturity

- **Level 0:** There is no difference between **testing and debugging**
- **Level 1:** The purpose of testing is to show **correctness**
- **Level 2:** The purpose of testing is to show that the software **does not work**
- **Level 3:** The purpose of testing is not to prove anything specific, but to **reduce the risk** of using the software
- **Level 4:** Testing is a **mental discipline** that helps all IT professionals develop higher quality software

Level 0 – Debug Only?

- **Level 0:** Testing is the same as debugging

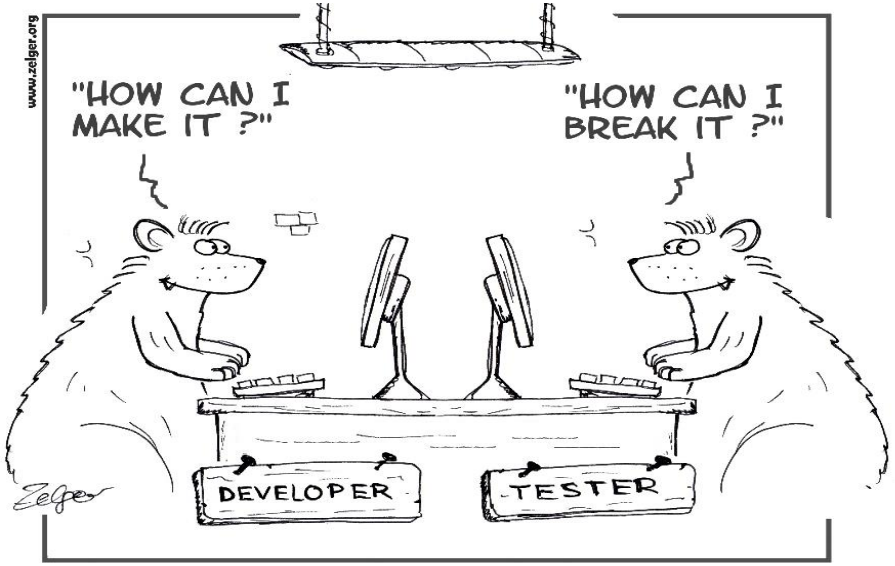
Not distinguish between incorrect behavior and mistakes in the program

Not help develop software that is reliable

In most CS programming classes, the students get their programs to compile, then debug the programs with a few inputs chosen either arbitrarily or provided by the professor

Levels 1, 2 – Developer vs Tester?

- **Level 1:** Purpose is to show correctness (developer-biased view)
  - Correctness is impossible to establish or demonstrate
  - Thus, test engineers usually have no strict goal, real stopping rule,
- **Level 2:** Purpose is to show failure (tester-biased view)
  - A negative view puts testers and developers into an adversarial relationship



They weren't so much different, but they had different goals

Levels 3, 4 – "Mature" Testing

- **Level 3**: Purpose is to show the presence of failures
  - There are risks when using software
  - Testers and developers cooperate to reduce risk
  - We see more and more companies move to this testing maturity level every year

- **Level 4:** Purpose is to increase quality of the software
  - Testing should be an integral part of the development process
  - Testers and developers cooperate to improve the software quality
  - Learn from mistake and improve software quality

How mature is your testing?

Are you at level 0, 1, 2, or 3?

We hopes to teach you to become "change agents" (level 4)

Summary

- Testing is the most time consuming and expensive part of software development
- Not testing is even more expensive
- Having too little testing effort early increases the testing cost
- Planning for testing after develop is prohibitively expensive
- A tester's goal is to **eliminate faults as early as possible**
- Testing improves **software quality**, **reduce cost**, and **preserve customer satisfaction**

What's Next?

- Fault, error, failure revisit
- Reachability, Infection, Propagation, and Revealability (RIPR) model

Questions?