Software Testing and Quality Assurance

Module-1

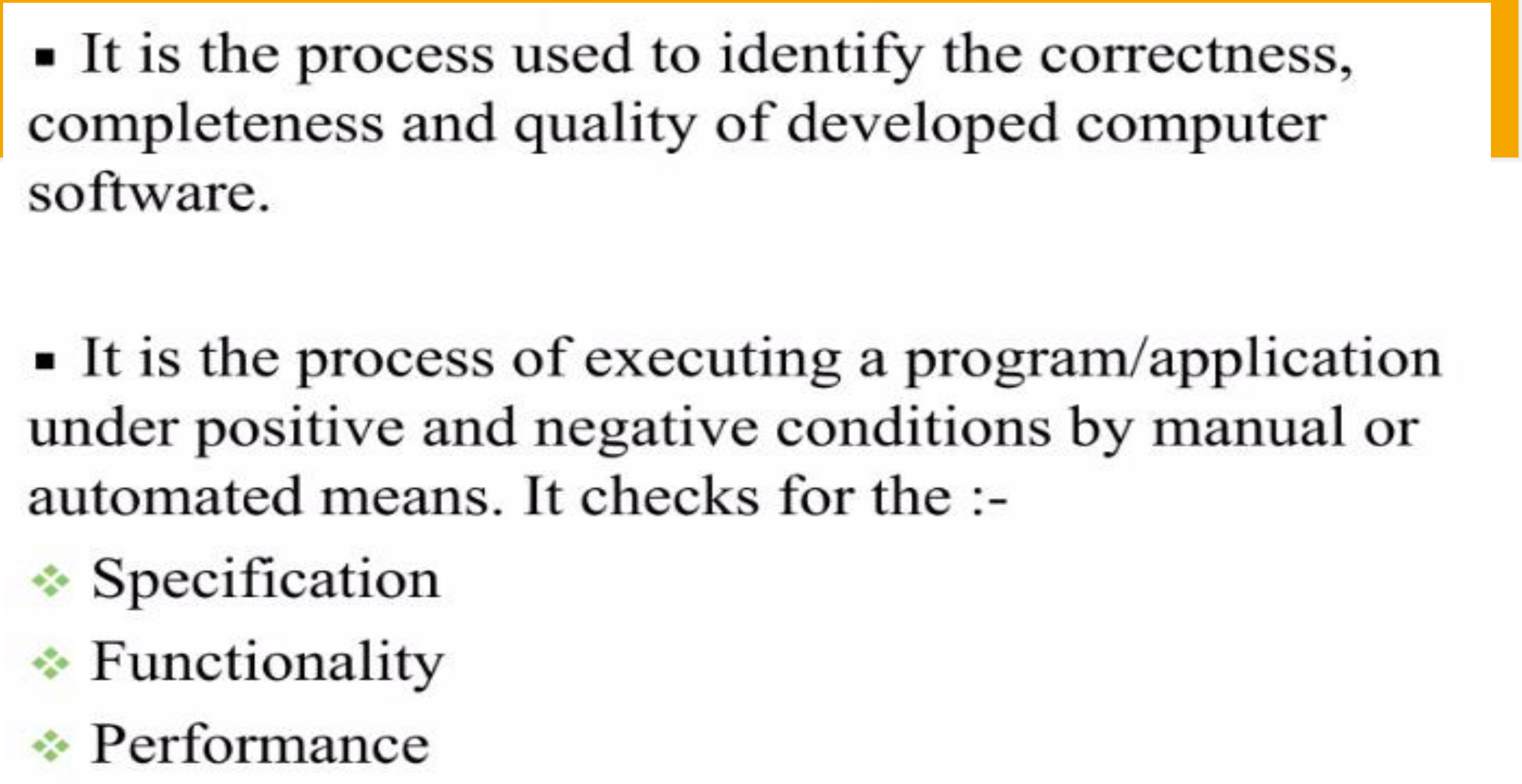
By Prof. Pallavi Mahajan

Contains

- Introduction to Software Testing
- Software Quality
- Role of Software Testing
- Verification and Validation
- Objectives and issues of testing
- Testing activities and levels
- Sources of Information for Test Case Selection

Introduction to Software Testing

What is Software Testing?



- It is the process used to identify the correctness, completeness and quality of developed computer software.

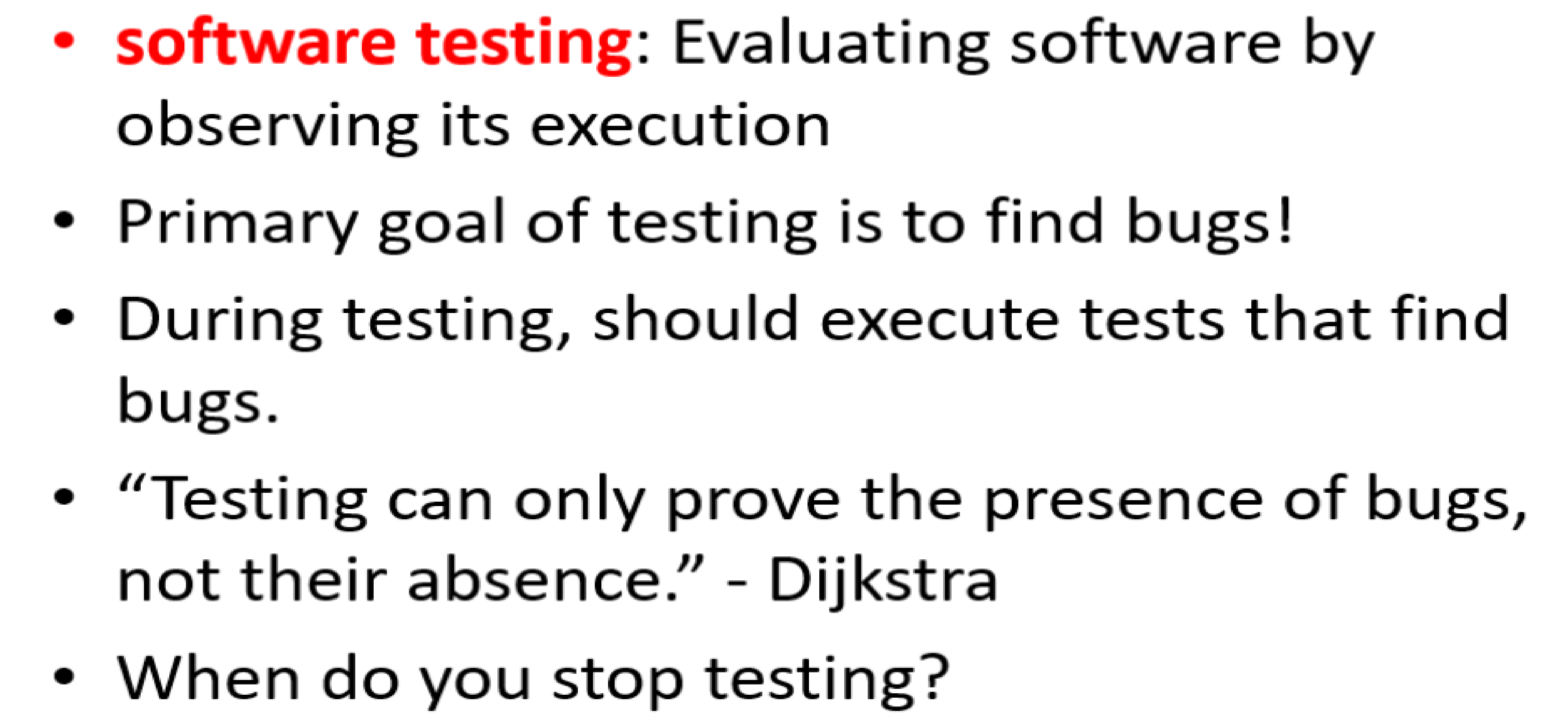
- It is the process of executing a program/application under positive and negative conditions by manual or automated means. It checks for the :-

❖ Specification

❖ Functionality

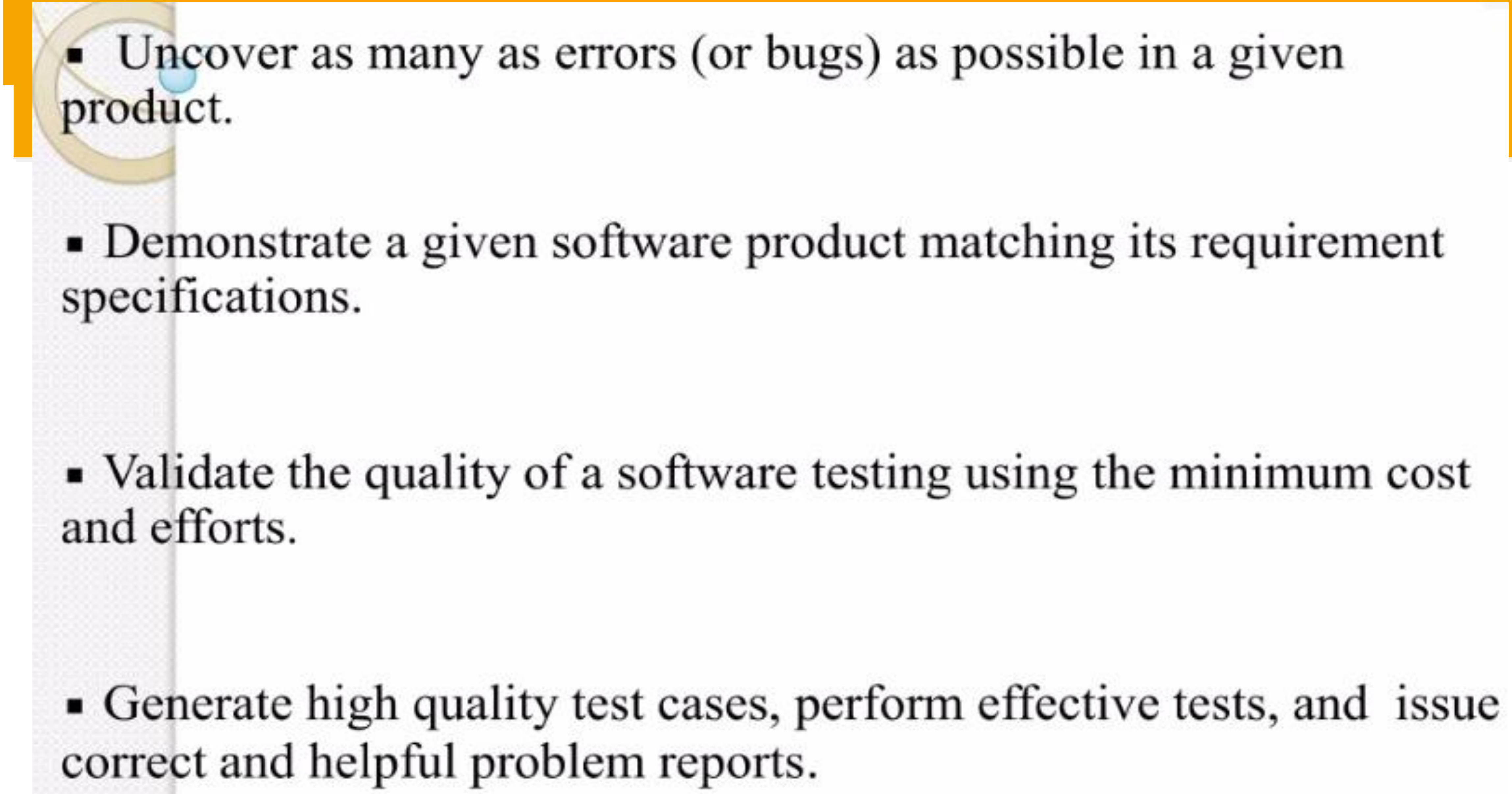
❖ Performance

What is Software Testing?



- **software testing**: Evaluating software by observing its execution
- Primary goal of testing is to find bugs!
- During testing, should execute tests that find bugs.
- “Testing can only prove the presence of bugs, not their absence.” - Dijkstra
- When do you stop testing?

Objectives of Software Testing



- Uncover as many as errors (or bugs) as possible in a given product.
- Demonstrate a given software product matching its requirement specifications.
- Validate the quality of a software testing using the minimum cost and efforts.
- Generate high quality test cases, perform effective tests, and issue correct and helpful problem reports.

Why Do We Test Software?

Testing in the 21st Century

- Software defines behavior
  - network routers, finance, switching networks, other infrastructure
- Today's software market :
  - is much bigger
  - is more competitive
  - has more users

Industry is going through a revolution in what testing means to the success of software products

- Embedded Control Applications
  - airplanes, air traffic control
  - spaceships
  - watches
  - ovens
  - remote controllers
  - PDAs
  - memory seats
  - DVD players
  - garage door openers
  - cell phones
- Agile processes put increased pressure on testers
  - Programmers must unit test – with no training or education!
  - Tests are key to functional requirements – but who builds those tests ?

Software is a Skin that Surrounds Our Civilization

Quote due to Dr. Mark Harman

Software Faults, Errors & Failures

- Software Fault : A static defect in the software
- Software Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior
- Software Error : An incorrect internal state that is the manifestation of some fault

Faults in software are equivalent to design mistakes in hardware.

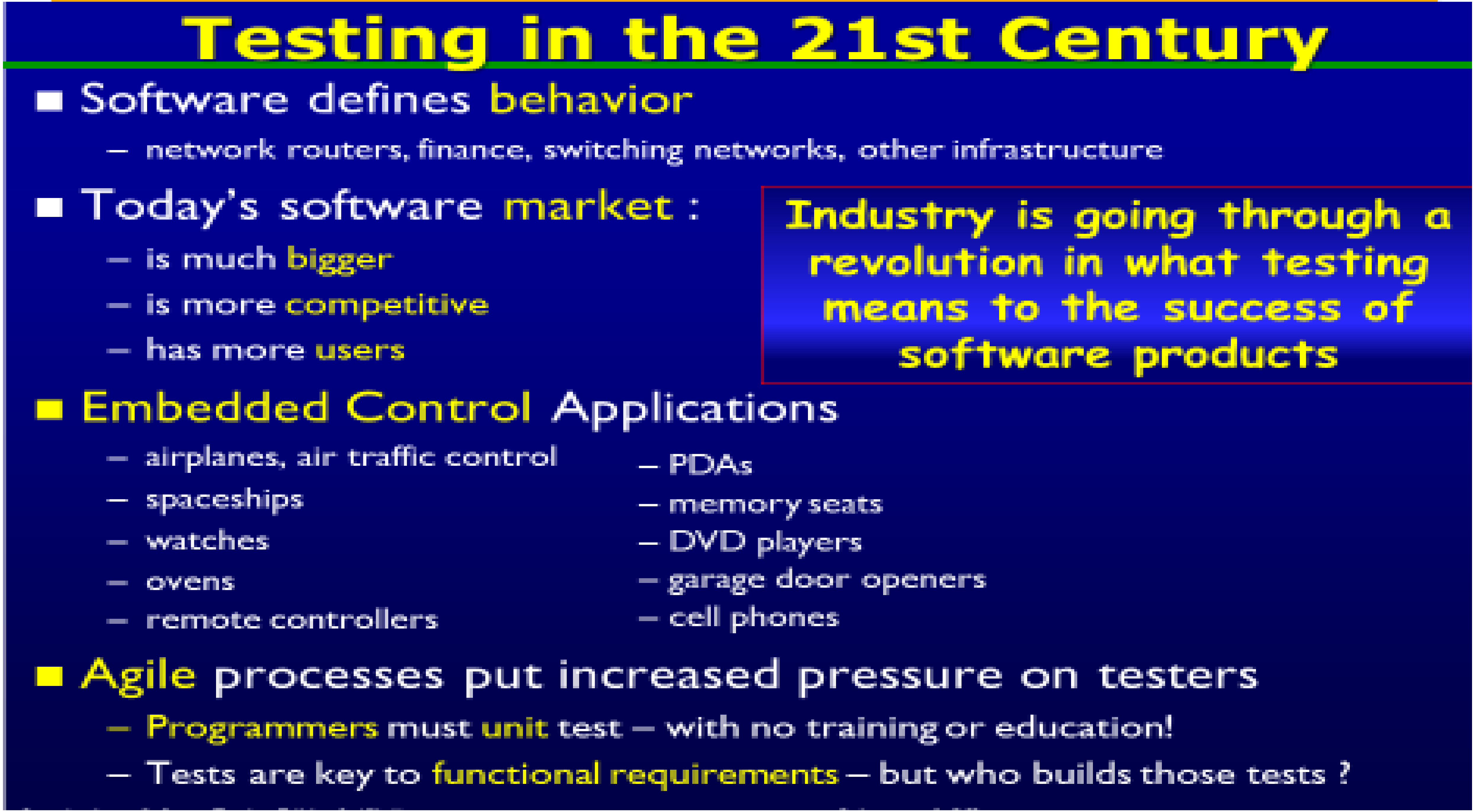
Software does not degrade.

Fault and Failure Example

- A patient gives a doctor a list of symptoms
  - Failures
- The doctor tries to diagnose the root cause, the ailment
  - Fault
- The doctor may look for anomalous internal conditions (high blood pressure, irregular heartbeat, bacteria in the blood stream)
  - Errors

Most medical problems result from external attacks (bacteria, viruses) or physical degradation as we age.

Software faults were there at the beginning and do not "appear" when a part wears out.



Testing in the 21st Century

- Software defines behavior
  - network routers, finance, switching networks, other infrastructure
- Today's software market :
  - is much bigger
  - is more competitive
  - has more users
- Embedded Control Applications
  - airplanes, air traffic control
  - spaceships
  - watches
  - ovens
  - remote controllers
  - PDAs
  - memory seats
  - DVD players
  - garage door openers
  - cell phones
- Agile processes put increased pressure on testers
  - Programmers must unit test – with no training or education!
  - Tests are key to functional requirements – but who builds those tests ?

Industry is going through a revolution in what testing means to the success of software products

Software is a Skin that Surrounds Our Civilization

Quote due to Dr. Mark Harman

Software Faults, Errors & Failures

- Software Fault : A static defect in the software
- Software Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior
- Software Error : An incorrect internal state that is the manifestation of some fault

Faults in software are equivalent to design mistakes in hardware.

Software does not degrade.

Failure, Error, Fault

- ***Failure:*** A failure is said to occur whenever the external behaviour of a system does not conform to that prescribed in the system specification.
- ***Error:*** An error is a state of the system. In the absence of any corrective action by the system, an error state could lead to a failure which would not be attributed to any event subsequent to the error.
- ***Fault:*** A fault is the adjudged cause of an error.

Fault and Failure Example

- A patient gives a doctor a list of symptoms
  - Failures
- The doctor tries to diagnose the root cause, the ailment
  - Fault
- The doctor may look for anomalous internal conditions (high blood pressure, irregular heartbeat, bacteria in the blood stream)
  - Errors

Most medical problems result from external attacks (bacteria, viruses) or physical degradation as we age.

Software faults were there at the beginning and do not “appear” when a part wears out.

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Error: i is 1, not 0, on the first iteration
Failure: none

Test 2
[0, 2, 7]
Expected: 1
Actual: 0

Error: i is 1, not 0
Error propagates to the variable count
Failure: count is 0 at the return statement

The Term Bug

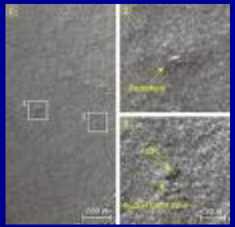
- *Bug* is used informally
- Sometimes speakers mean fault, sometimes error, sometimes failure … often the speaker doesn’t know what it means !
- This class will try to use words that have precise, defined, and unambiguous meanings

“It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise—this thing gives out and *[it is]* then that 'Bugs'—as such little faults and difficulties are called—show themselves and months of intense watching, study and labor are requisite. . .” – Thomas Edison

“an analyzing process must equally have been performed in order to furnish the Analytical Engine with the necessary operative data; and that herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, the cards may give it wrong orders. ” – Ada, Countess Lovelace (notes on Babbage’s Analytical Engine)

Spectacular Software Failures

- NASA's Mars lander: September 1999, crashed due to a units integration fault



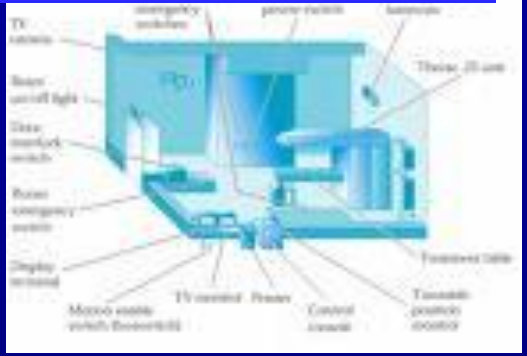
Mars Polar Lander crash site?

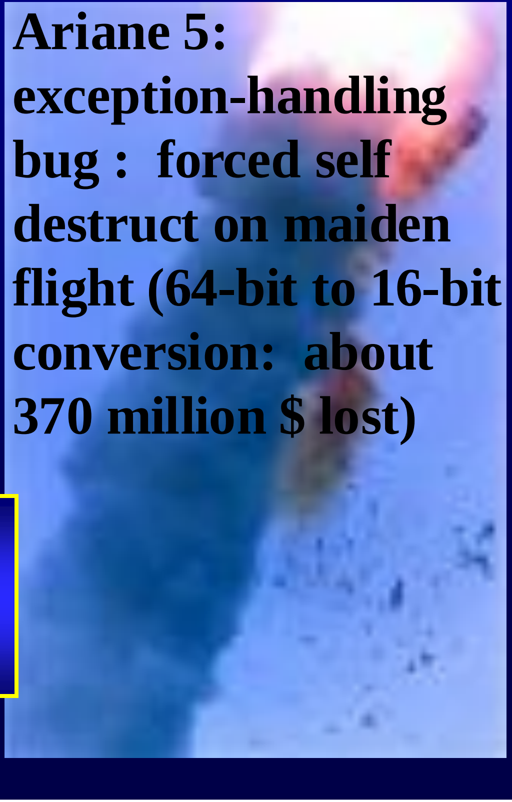
- THERAC-25 radiation machine : Poor testing of safety-critical software can cost *lives* : 3 patients were killed
- Ariane 5 explosion : Millions of $$
- Intel's Pentium FDIV fault : Public relations nightmare

We need our software to be dependable

Testing is *one* way to assess dependability

THERAC-25 design





Ariane 5: exception-handling bug : forced self destruct on maiden flight (64-bit to 16-bit conversion: about 370 million $ lost)

Northeast Blackout of 2003

508 generating units and 256 power plants shut down

Affected 10 million people in Ontario, Canada

Affected 40 million people in 8 US states

Financial losses of $6 Billion USD

The alarm system in the energy management system failed due to a software error and operators were not informed of the power overload in the system

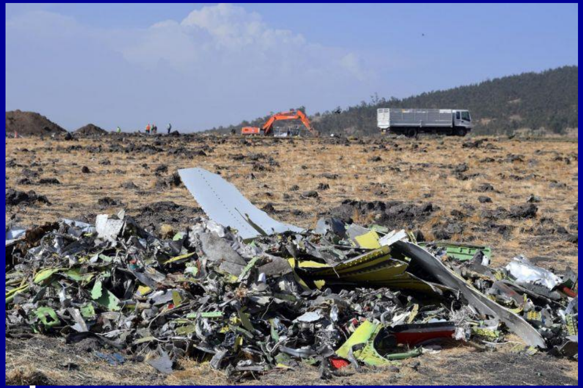
Costly Software Failures

- NIST report, “The Economic Impacts of Inadequate Infrastructure for Software Testing” (2002)
  - Inadequate software testing costs the US alone between $22 and $59 billion annually
  - Better approaches could cut this amount in half
- Huge losses due to web application failures
  - Financial services : $6.5 million per hour (just in USA!)
  - Credit card sales applications : $2.4 million per hour (in USA)
- In Dec 2006, *amazon.com’s* BOGO offer turned into a double discount
- 2007 : Symantec says that most security vulnerabilities are due to faulty software

World-wide monetary loss due to poor software is staggering

Spectacular software Failures

- Boeing A220 : Engines failed after software update allowed excessive vibrations



- Boeing 737 Max : Crashed due to overly aggressive software flight overrides (MCAS)

- Toyota brakes : Dozens dead, thousands of crashes

- Healthcare website : Crashed repeatedly on launch—never load tested

- Northeast blackout : 50 million people, $6 billion USD lost … alarm system failed

Software testers try to find faults before the faults find users

Testing in the 21st Century

- More safety critical, real-time software
- Embedded software is ubiquitous … check your pockets
- Enterprise applications means bigger programs, more users
- Paradoxically, free software increases our expectations !
- Security is now all about software faults
  - Secure software is reliable software
- The web offers a new deployment platform
  - Very competitive and very available to more users
  - Web apps are distributed
  - Web apps must be highly reliable

Industry desperately needs our inventions !

The True Cost of Software Failure

Fail watch analyzed news articles for 2016

- 606 reported software failures
- Impacted half the world's population
- Cost a combined $1.7 trillion US dollars

Poor software is a significant drag on the world's economy

Not to mention frustrating

What Does This Mean?

Software testing is getting more important

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

Validation & Verification (*IEEE*)

- Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage

- 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

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

Testing Goals Based on Test Process Maturity

- Level 0 : There's 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 doesn't 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 Thinking

- Testing is the same as debugging
- Does not distinguish between incorrect behavior and mistakes in the program
- Does not help develop software that is reliable or safe

This is what we teach undergraduate CS majors

Level 1 Thinking

- Purpose is to show correctness
- Correctness is impossible to achieve
- What do we know if no failures?
  - Good software or bad tests?
- Test engineers have no:
  - Strict goal
  - Real stopping rule
  - Formal test technique
  - Test managers are powerless

This is what hardware engineers often expect

Level 2 Thinking

- Purpose is to show failures
- Looking for failures is a negative activity
- Puts testers and developers into an adversarial relationship
- What if there are no failures?

This describes most software companies.

How can we move to a *team approach* ??

Level 3 Thinking

- Testing can only show the presence of failures
- Whenever we use software, we incur some risk
- Risk may be small and consequences unimportant
- Risk may be great and consequences catastrophic
- Testers and developers cooperate to reduce risk

This describes a few “enlightened” software companies

Level 4 Thinking

A mental discipline that increases quality

- Testing is only one way to increase quality
- Test engineers can become technical leaders of the project
- Primary responsibility to measure and improve software quality
- Their expertise should help the developers

This is the way "traditional" engineering works

Where Are You?

Are you at level 0, 1, or 2 ?

Is your organization at work at level 0, 1, or 2 ?

Or 3?

We hope to teach you to become "change agents" in your workplace …

Advocates for level 4 thinking

Tactical Goals : Why Each Test ?

If you don't know why you're conducting each test, it won't be very helpful

- Written test objectives and requirements must be documented
- What are your planned coverage levels?
- How much testing is enough?
- Common objective – spend the budget … test until the ship-date …
  - Sometimes called the "date criterion"

Here! Test This!

Offutt’s first “professional” job

A stack of computer printouts—and no documentation

Why Each Test ?

If you don't start planning for each test when the functional requirements are formed, you'll never know why you're conducting the test

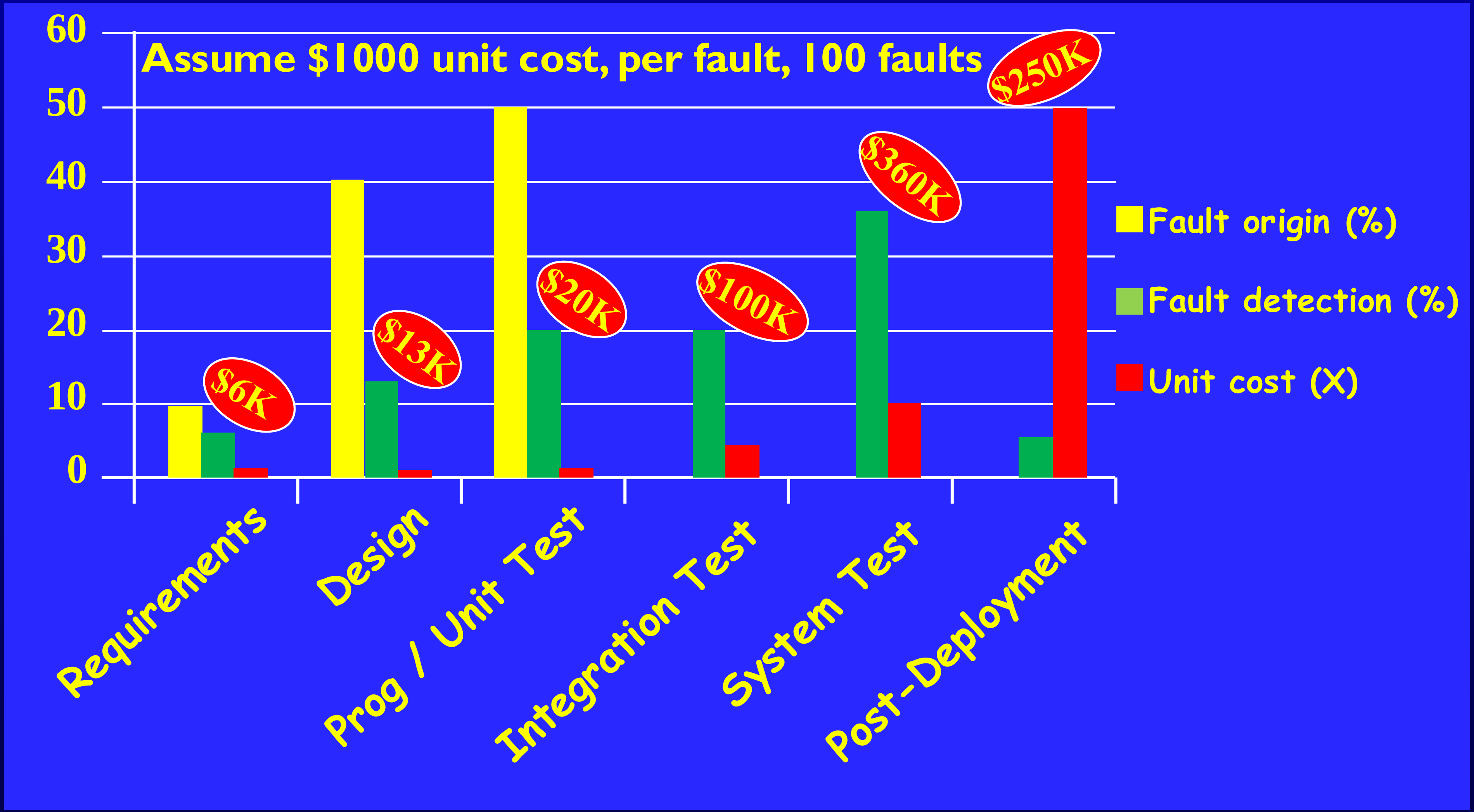
- 1980: "The software shall be easily maintainable"
- Threshold reliability requirements?
- What fact does each test try to verify?
- Requirements definition teams need testers!

Cost of Not Testing

Poor Program Managers might say:
"Testing is too expensive."

- Testing is the most time consuming and expensive part of software development
- Not testing is even more expensive
- If we have too little testing effort early, the cost of testing increases
- Planning for testing after development is prohibitively expensive

Cost of Late Testing



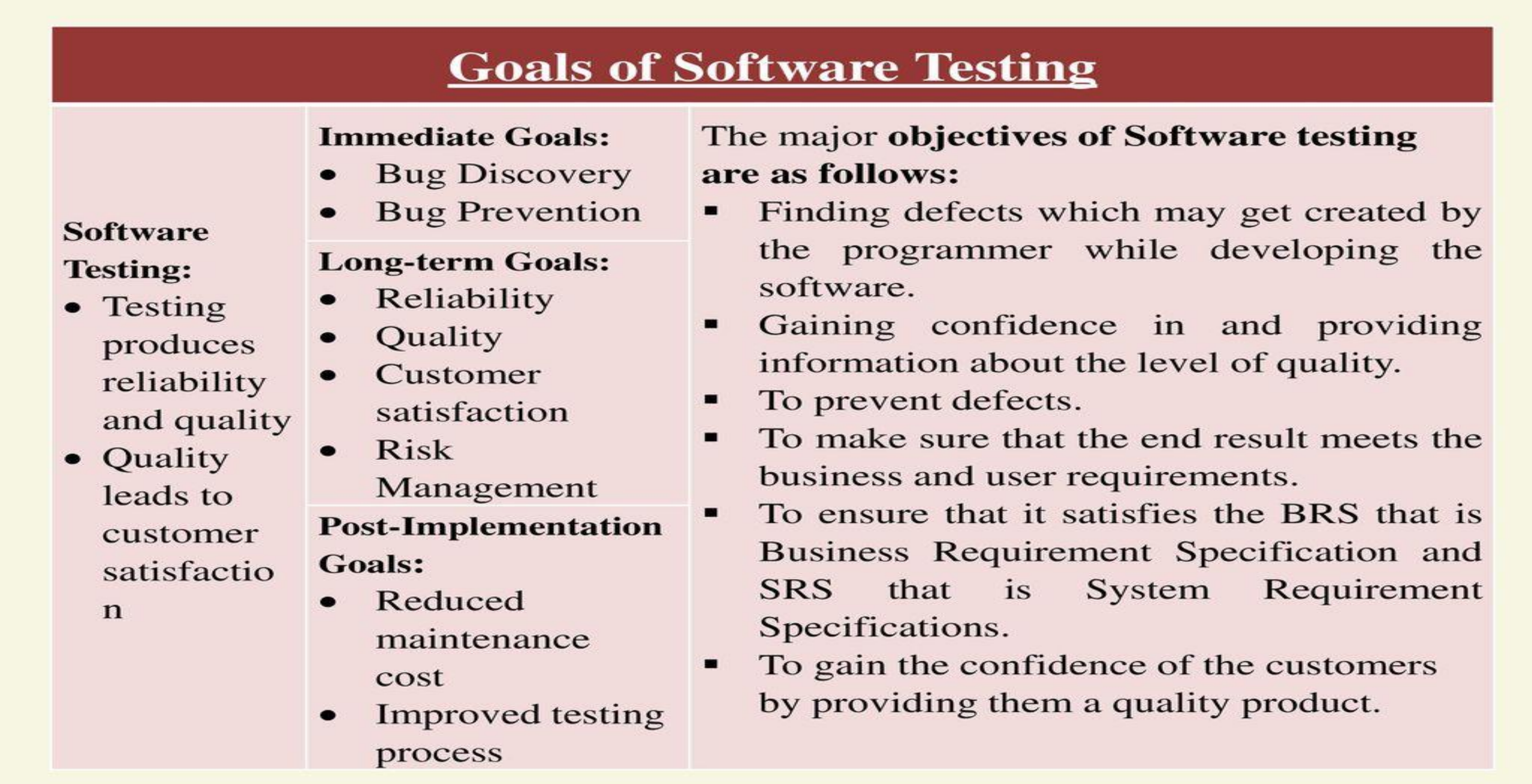
Software Engineering Institute; Carnegie Mellon University; Handbook CMU/SEI-96-HB-002

Summary: Why Do We Test Software ?

A tester's goal is to eliminate faults as early as possible

- Improve quality
- Reduce cost
- Preserve customer satisfaction

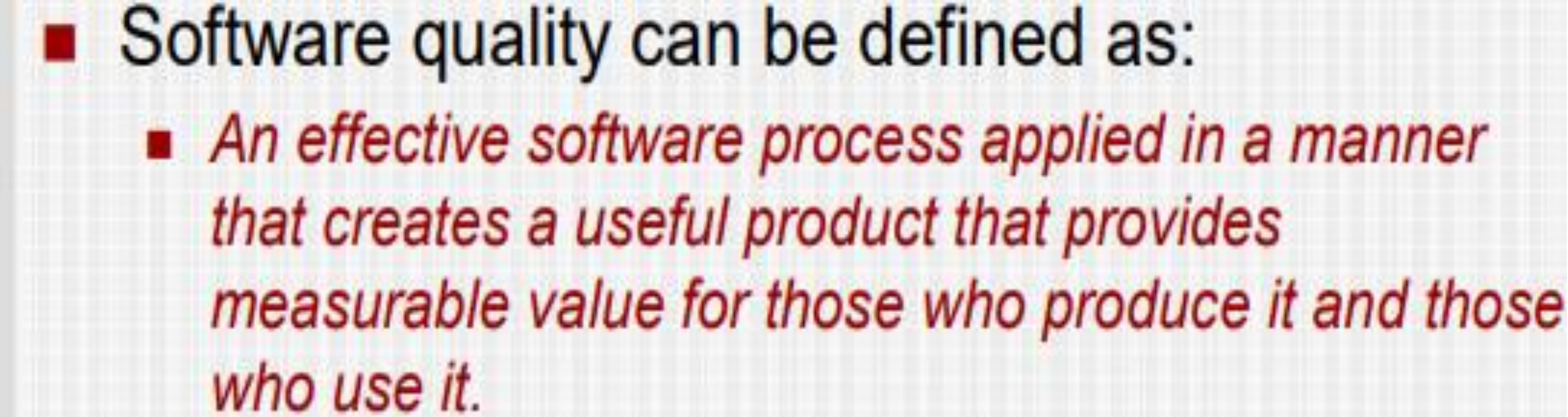
Goals of Software Testing



Goals of Software Testing		
Software Testing: • Testing produces reliability and quality • Quality leads to customer satisfaction	**Immediate Goals:** • Bug Discovery • Bug Prevention	The major **objectives of Software testing are as follows:** ▪ Finding defects which may get created by the programmer while developing the software. ▪ Gaining confidence in and providing information about the level of quality. ▪ To prevent defects. ▪ To make sure that the end result meets the business and user requirements. ▪ To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications. ▪ To gain the confidence of the customers by providing them a quality product.
	Long-term Goals: • Reliability • Quality • Customer satisfaction • Risk Management	
	Post-Implementation Goals: • Reduced maintenance cost • Improved testing process	

Software Quality

Software Quality- Definition



- Software quality can be defined as:
  - *An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.*

What is Software Quality?

- Software quality is a complex concept with multiple perspectives.
- It varies across domains such as philosophy, economics, marketing, and management.
- Different experts have explored software quality through various models and views.

Five Views of Software Quality

1. **Transcendental View:** Difficult to define but recognizable.
2. **User View:** Quality is fitness for purpose—does it meet user needs?
3. **Manufacturing View:** Quality is conformance to specifications.
4. **Product View:** Quality is tied to inherent characteristics of the product.
5. **Value-Based View:** Quality depends on customer willingness to pay.

Software Quality Attributes

- Modularity
- Complexity
- Portability
- Usability
- Reusability
- Efficiency
- Learnability

- Safety
- Security
- Reliability
- Resilience
- Robustness
- Understandability
- Testability
- Adaptability

Historical Background of Software Quality

- **McCall, Richards, and Walters** (1970s) introduced quality factors and criteria.
- **Quality Factors:** Correctness, reliability, efficiency, maintainability, etc.
- **Quality Criteria:** Attributes related to development processes (e.g., modularity for maintainability).

Software Quality Models

- **ISO 9126**: Defines six categories of quality characteristics:
  - Functionality, reliability, usability, efficiency, maintainability, portability.
- **CMM (Capability Maturity Model):** Measures process maturity on a 5-level scale.
  - Levels from Initial (1) to Optimized (5).

Testing Process Models

- **TPI Model** and **TMM Model** help assess and improve software testing processes.

The Role of Testing

The Role of Testing in Software Quality

- Testing is key to both **improving** and **assessing** software quality.
- The test–find defects–fix cycle improves quality during development.
- System-level tests evaluate the overall quality before release.

Static vs Dynamic Analysis

- **Static Analysis:**
  - Involves reviewing documents (requirements, design, source code) without executing the code.
  - Includes code reviews, inspections, and algorithm analysis.
- **Dynamic Analysis:**
  - Involves executing the software to observe its behavior and performance.
  - Relies on testing with real or representative inputs to find defects.

Verification vs Validation

Verification and Validation in Software Testing

- **Verification:** Ensures the product meets requirements at each development phase.
- **Validation:** Ensures the product meets the user's needs and expectations.
- Verification confirms "Are we building the product correctly?”
- Validation confirms "Are we building the correct product?"

Verification vs Validation: Key Differences

- Verification: Focuses on reviewing intermediate products (e.g., design, code).
- Validation: Focuses on the final product through system testing in real environments.

Key Points

- Software quality is multi-faceted and influenced by various perspectives and models.
- Effective testing and clear verification/validation processes are essential to ensuring quality.
- Continuous improvement and evaluation are

Objectives of Testing

What Are the Objectives of Testing?

- Testing plays a critical role in ensuring the quality of software.
- The goals vary depending on the stakeholders, their perspectives, and the phase of development.
- These objectives help determine how testing is performed and what outcomes are expected.

Key Stakeholders in Testing

- **Programmers**: Test to ensure the system or unit works as intended.
- **Test Engineers**: Focus on finding defects and verifying system behavior.
- **Project Managers**: Concerned with the overall risk and cost of testing.
- **Customers**: Interested in the reliability and quality of the final product.

Objective 1: It Works—Confirming Basic Functionality

- **Programmers' Perspective**: Test if a unit or system works in normal conditions.
- **Psychological Aspect**: A key objective is to show that the system works rather than proving it doesn’t.
- Testing here provides confidence that basic functionality is achieved.

Objective 2: It Does Not Work–Finding Faults

- After ensuring functionality, the next objective is to find faults by making the system fail.
- **Programmers/Development Team's Perspective**: They try to break the system to identify issues and improve it.
- Tests are designed to push the system beyond normal operational conditions.

Objective 3: Reduce the Risk of Failure

- **Complex Systems**: Faults are inevitable in large, complex systems, and the goal is to minimize the risk of failure.
- As defects are discovered and fixed, the system's failure rate decreases.
- **Test Objective**: To reduce failure rates to an acceptable level, increasing overall system reliability

Objective 4: Reduce the Cost of Testing

- Testing incurs costs in multiple areas:
  - Designing, maintaining, and executing test cases.
  - Analyzing test results and documenting them.
  - Executing the system and generating reports.
- **Cost Optimization**: Aim to use fewer test cases while maintaining effectiveness.
- Focus on reducing unnecessary costs while ensuring low-risk software.

Key Points

- Testing objectives help guide the testing process toward producing reliable and efficient software.
- The four key objectives—validating functionality, identifying faults, reducing failure risk, and minimizing testing costs—must be balanced effectively.
- Proper stakeholder involvement and test case selection are critical to achieving these objectives.

Issues in Testing

Central Issue in Testing: The Challenge of Complete Testing

- The ideal outcome of testing is to discover all faults, but this is near-impossible.
- Instead, a subset of the input domain is tested, which may only exercise part of the program's behavior.
- The key challenge is selecting an appropriate subset of inputs to achieve accurate and meaningful test results.

Testing Activities

Key Activities in Testing

- There are several steps that a test engineer follows to conduct effective testing.
- These activities are crucial for performing structured and thorough testing.

What are the Testing Activities

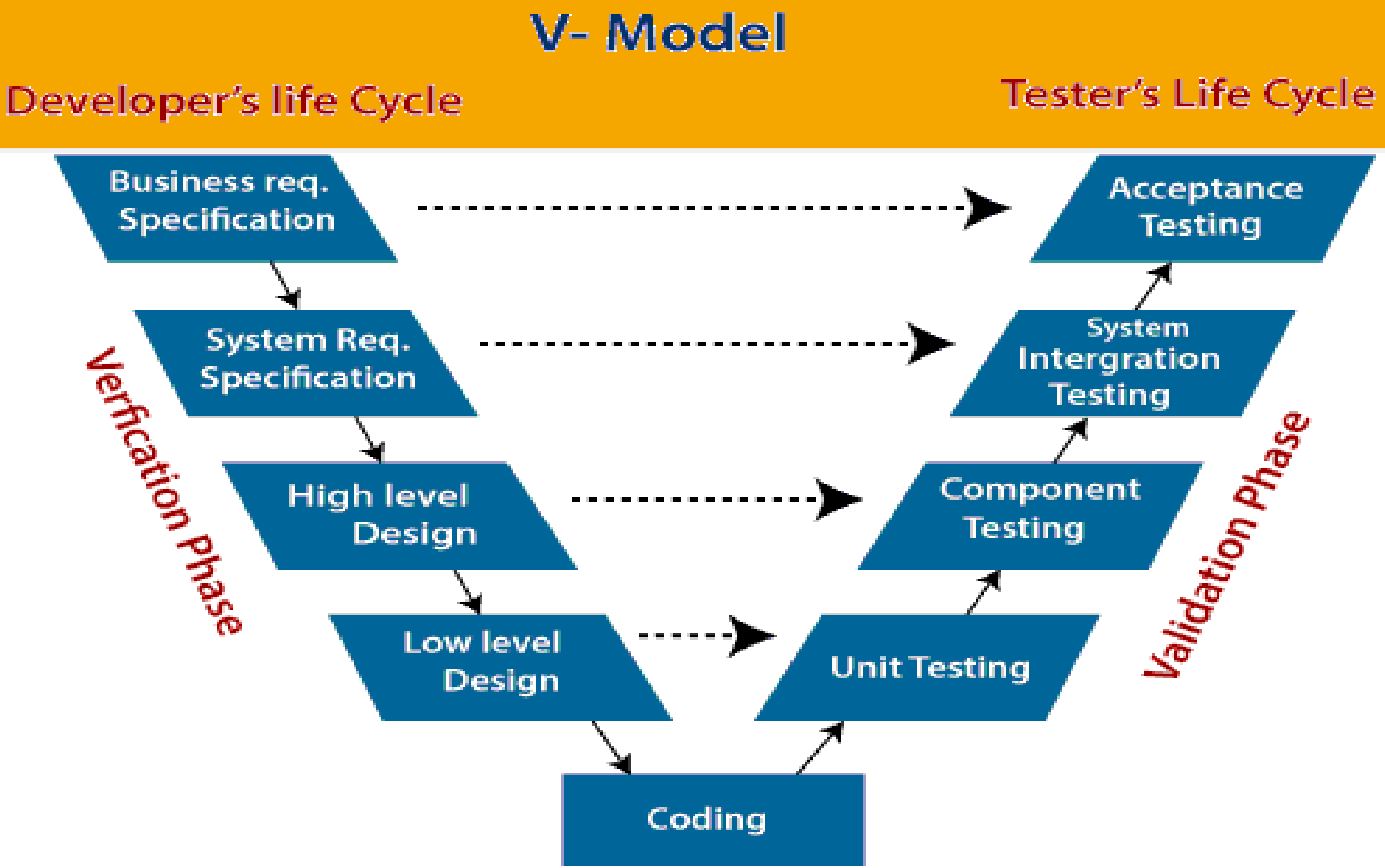
- Identify the Objective to be Tested
- Select Inputs
- Compute Expected Outcome
- Set Up the Execution Environment
- Execute the Program
- Analyze the Test Results
- Understanding Test Verdicts
- Test Reporting

Test Levels

Introduction to Test Levels

- Testing is performed at different levels in a software system's life cycle.
- There are four main stages of testing: **Unit Testing, Integration Testing, System Testing, and Acceptance Testing**.
- These levels ensure the software functions correctly and meets customer requirements.

The Classical V-Model



The Classical V-Model

- **V-Model** represents the development and testing phases in software engineering.
- The model emphasizes parallelism between development and testing.
- **Stages:**
  - Unit Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing

Unit Testing

- **Definition:** Testing of individual units (functions, methods, classes) in isolation.
- **Performed by:** Programmers.
- **Objective:** Ensure that individual components work as expected.
- **Focus:** Isolated testing of specific units of code.

Integration Testing

- **Definition:** Combining individual units and testing them as a group.
- **Performed by:** Developers and integration test engineers.
- **Objective:** Ensure that modules work together as expected.
- **Key Consideration:** Ensuring stability before system-level testing.

System Testing

- **Definition:** Testing the complete system to verify that it meets requirements.
- **Types of Tests:**
  - Functionality, security, robustness, load, stability, stress, performance, reliability.
- **Objective:** Identify defects and ensure system readiness.
- **Critical Phase:** Occurs close to delivery, focuses on defect discovery.

Regression Testing

- **Definition:** Testing to ensure that changes (modifications or fixes) haven’t introduced new defects.
- **Performed during:** Unit, integration, and system testing.
- **Objective:** Validate that existing features are unaffected by changes.

Regression Testing Process

- **Test Case Selection:** No new tests are created; existing tests are executed.
- **Focus:** Ensure no new faults are introduced after modifications.
- **Cost:** Regression testing accounts for a large portion of testing effort.
- **Challenge:** Prioritizing test cases to uncover new faults effectively.

Sources of Information for Test Case Selection

Key Sources for Test Case Selection

- Requirements and functional specifications.
- Source code.
- Input and output domains.
- Operational profiles.
- Fault models.

Requirements & Functional Specifications

- Source of test cases based on user needs and system functionality.
- **Informal Examples:** Text descriptions, flowcharts, use cases.
- **Formal Examples:** Finite state machines, Z notation.
- **Purpose:** Ensure software meets functional expectations.

Source Code

- Provides the actual behavior of the software system.
- Test cases are designed based on the code’s structure and implementation.
- **Example:** A sorting algorithm might require tests based on its specific implementation.

Input & Output Domains

- **Input & Output Domains**
- Special cases in input and output need careful consideration.
- **Example:** Factorial function requires handling edge cases like n=0.
- Key focus on values like 0 and 1 which have special mathematical properties.

Operational Profiles

- Quantitative characterization of how a system will be used.
- **Purpose:** Guide test case selection based on actual usage patterns.
- Example: Testing web applications based on real user session data.

Fault Models

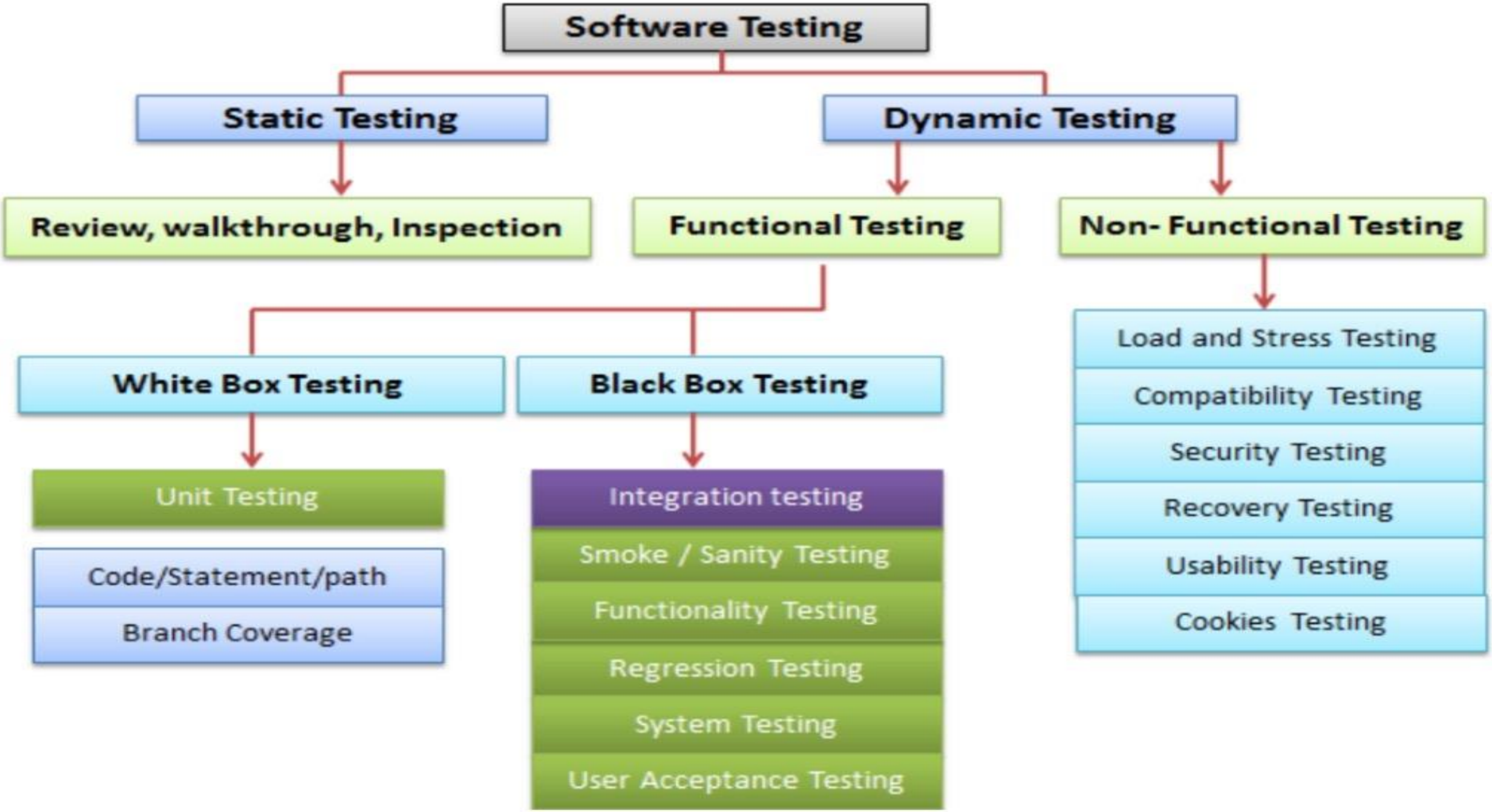
- Previous faults provide insight into potential future errors.
- Types of fault-based testing:
  - **Error Guessing:** Using experience to guess where faults might be.
  - **Fault Seeding:** Injecting faults to assess test suite effectiveness.
  - **Mutation Analysis:** Making slight changes to see if test cases catch errors.

Key Points

- Four main stages of testing: **Unit Testing, Integration Testing, System Testing, and Acceptance Testing**.
- Regression testing ensures modifications don’t introduce new defects.
- Effective test case selection is based on multiple sources like requirements, code, and fault models.

Thank You..

Types of Software Testing:



Software Testing

Static Testing

Dynamic Testing

Review, walkthrough, Inspection

Functional Testing

Non- Functional Testing

White Box Testing

Black Box Testing

Unit Testing

Code/Statement/path

Branch Coverage

Integration testing

Smoke / Sanity Testing

Functionality Testing

Regression Testing

System Testing

User Acceptance Testing

Load and Stress Testing

Compatibility Testing

Security Testing

Recovery Testing

Usability Testing

Cookies Testing