Software Testing

Like Death and Taxes, Testing is both Unpleasant and Inevitable - Ed Yourdon

By:
Dr. Vinod L Desai,
Associate Professor,
NICSM (Formerly Known as SVICS),
MCA College, Kadi-382715.
Email: vinodl_desai@yahoo.com

Software Testing is the process of executing a program or system with the intent of finding errors.

- Software is not like other physical processes
- Where software differs is in the manner in which it fails
- Most physical systems fail in a fixed (and reasonably small) set of ways
- By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible

Strategic Approach to Testing - 1

- Testing begins at the component level and works outward toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software conducts testing and may be assisted by independent test groups for large projects.
- The role of the independent tester is to remove the conflict of interest inherent when the builder is testing his or her own product.

Strategic Approach to Testing - 2

- Testing and debugging are different activities.
- Debugging must be accommodated in any testing strategy.
- Need to consider verification issues
  - are we building the product right?
- Need to Consider validation issues
  - are we building the right product?

Testing is NOT

- Code inspections
- Design reviews
- Configuration management
- Bug tracking

- These are, along with **testing**, are part of Software Quality Assurance (SQA).
- Together these improve the quality of the product

Testing Life Cycle

- Establish test objectives.
- Design criteria (review criteria).
  - Correct.
  - Feasible.
  - Coverage.
  - Demonstrate functionality.
- Writing test cases.
- Testing test cases.
- Execute test cases.
- Evaluate test results.

Who Tests?

- Developers
  - Unit testing
- QA Team
  - Module testing, automated testing
- Software
  - Automated testing
- Designers
  - Engineers, Analysts
- Users
  - Unpleasant but true

Software Testing Principles

- All tests should be traceable to customer requirements.
- Tests should be planned long before testing begins.
- 80% of all errors uncovered (revealed) during testing will likely be traceable to 20% of all program modules.
- Testing should begin “in the small” and progress toward testing “in the large”.
- Exhaustive testing is not possible.
- To be most effective, testing should be conducted by an independent third party.

Software Testability

According to James Bach:

- Software testability is simply how easily a computer program can be tested.
- **Operability:** "the better it works, the more efficiently it can be tested."
- **Observability:** "What you see is what you test."
- **Controllability:** "The better we can control the software, the more the testing can be automated and optimized."
- **Decomposability:** "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting."
- **Simplicity:** "The less there is to test, the more quickly we can test it."
- **Stability:** "The fewer the changes, the fewer the disruptions to testing."
- **Understandability:** "The more information we have, the smarter we will test."

Testing Process Goals

- Validation Testing
  - To demonstrate to the developer and the system customer that the software meets its requirements;
  - A successful test shows that the system operates as intended.
- Defect Testing
  - To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification;
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

Why Do We Test?

- Provide confidence in the system
- Identify areas of weakness
- Establish the degree of quality
- Establish the extent that the requirements have been met, i.e. what the users asked for is what they got not what someone else though they wanted
- To provide an understanding of the overall system
- To prove it is both usable and operable
- To provide sufficient information to allow an objective decision on applicability to deploy

Another view – cynical?

- To use up spare budget
- To provide a good excuse why the project is late
- To provide jobs for people who can't code
- To make the developers look good
- To provide the project manager with some contingency in the plan and can be cut if push come to throw!

Are we capable of testing?

- Is an experienced tester better than an experienced user at finding faults?

- How can testers help themselves and users?
  - Working with the users to understand their systems
  - Providing testing skills transfer
  - Attending testing industry conferences
  - Attaining industry recognised software testing qualifications

When?

- Once the code is complete?
- As soon as the architecture is defined?
- Once the system delivery is complete?
- During development?
- Once the business requirements have been defined?
- As soon as the project is given the go-ahead?

Error, Defect and Bug

- Error-Mistakes in code of application is error due to errors in coding.
- Defects-Mismatches found by testers during testing of the application build are termed as defects.
  Mismacthes three types
  1. Mismatch between expected and actual
  2. Missing functionality with respect to customer requirements
  3. Extra functionality with respect to customer requirements.
- Bugs-If defects are accepted by Developers to be solved are bugs.

  Note:
  some defects cannot be accepted as bugs by developers as they may be coming in future versions of the application

Continue…

- Error: It is deviation from logic, syntax or execution. 3 types:

  Syntactical error: it is deviation from syntax of program

  Logical error: It is deviation from logic of the program.

  Executional error: This occurs during executing the program.
- Defect: When the error is discovered by the tester then it is called defect. E.g. if we double click on any Folder existing on the root node and get an Internal Error or System Crash message then its a Defect.
- Bug: If the developer accepts the defect then it is called bug.

  E.g. If something is already documented to be implemented and implementation is not consistent as per requirement then its a BUG.
- Error - A Fault in the program leads to error in program execution.
- Defect- If the error caught by tester then it is called as defect.
- Bug - If the defect is accepted by the development people then it is called as Bug.
- Fault---->Error---->Defect---->Bug

Errors

- A variation from the *expected* often becomes an error.
- Errors are a part of life. The process for their creation is in-built into nature by nature.
- They exist for anyone who has the ability to observe.

Error: Elimination or Reduction?

- In most practical situations, total error elimination is a myth.
- Error reduction based on the economics of software development is a practical approach.

Errors:Consequences

- An error might lead to a failure.
- The failure might cause a minor inconvenience or a catastrophe.
- The complexity of an error has no known correlation with the severity of a failure. *The "misplaced break" is an example of a simple error that caused the AT&T phone-jam in 1990.*

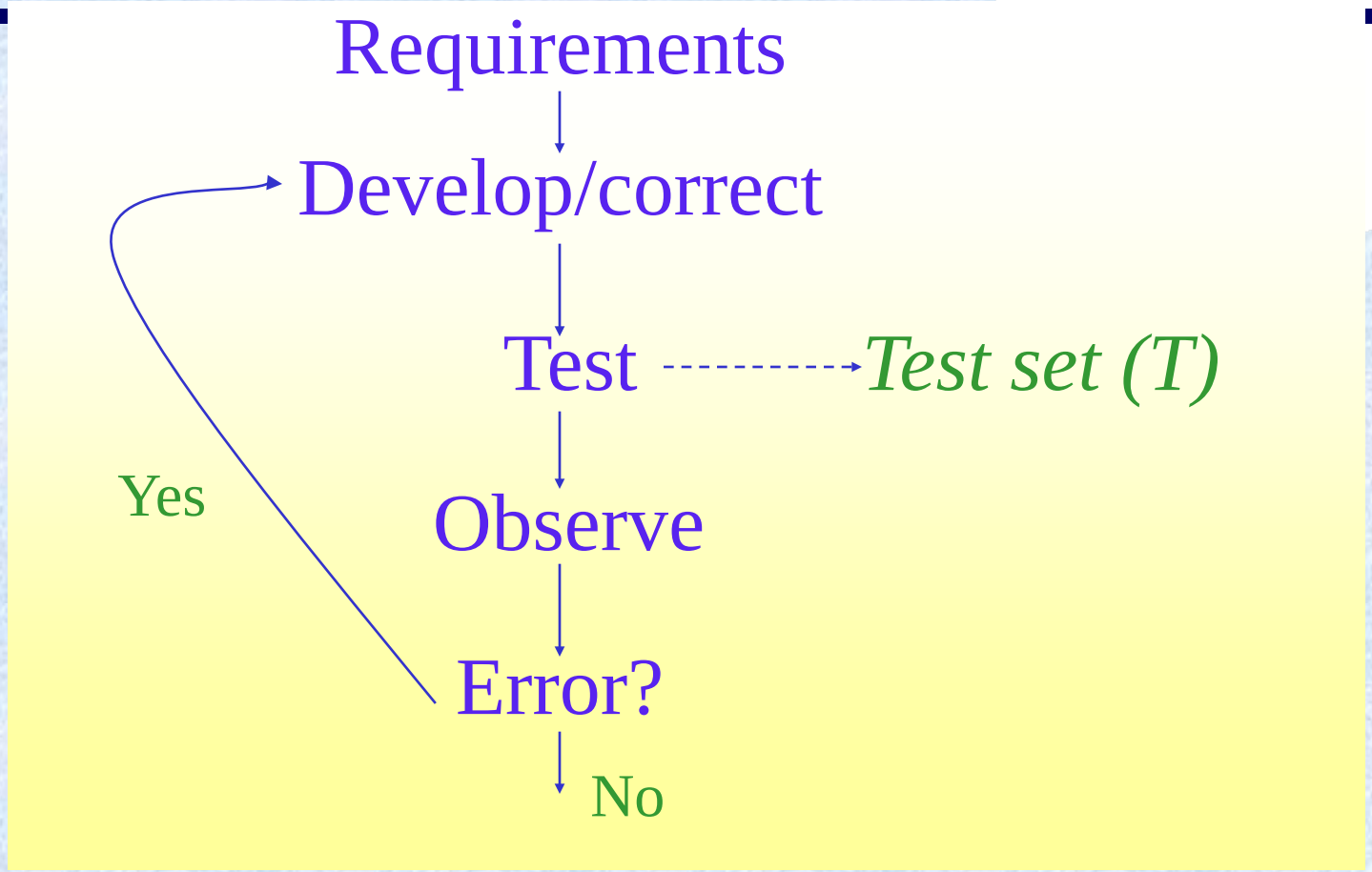
Errors:Unavoidable!

- Errors are bound to creep into software.
- This belief enhances the importance of testing.
- Errors that creep in during various phases of development can be removed using a well defined and controlled process of software testing.

Errors:Probability

- The probability of a program delivered with errors can be reduced to a small quantity.....but not to 0!
- Exceptions to the above can be created with the help of programs that have a finite input domain.
- Verification and inspection help reduce errors and are complementary to testing.

Error Detection and Removal



Errors: Examples

- Windows 95: “large” error database maintained by Microsoft (proprietary)
- Several other error studies published.
- Error studies have also been published in other diverse fields such as in music, speech, sports, and civil engineering.

Infamous Software Error Case Studies

- **Intel Pentium Floating-Point Division Bug, 1994**

  (4195835 / 3145727) * 3145727 – 4195835

  If the answer is zero, your computer is just fine.

  If you get anything else, you have an old Intel Pentium CPU with floating-point division bug – a software bug burned into computer chip.

  What makes this story notable isn't the bug, but the way Intel handled the situation.

  Their software test engineers had found the problem while performing their own tests before the chip was released.

- Intel management decided that the problem was not severe enough or likely to warrant fixing it, or even publicizing it.
- Once the bug was found, Intel attempted to diminish its perceived severity through press releases and public statements.
- When pressured, Intel offered to replace the faulty chips, but only if a user could prove that he was affected by the bug.

Internet newsgroups were jammed with irate customers demanding that Intel fix the problem.

Intel apologized for the way it handled the bug and took a charge over $400 million to cover the cost.

Patriot Missile Defense System, 1991

The U.S. Patriot missile defense system is a scaled back version of the “Star Wars” program proposed by President Ronald Reagan.

It was first put to use in the Gulf War as a defense for Iraqi Scud missiles.

Although there were many news stories touting the success of the system, it did fail to defend against several missiles, including one that killed 28 U.S. soldiers in Dhahran.

Analysis found that a software bug was the problem.

Immature Versus Mature Software Processes

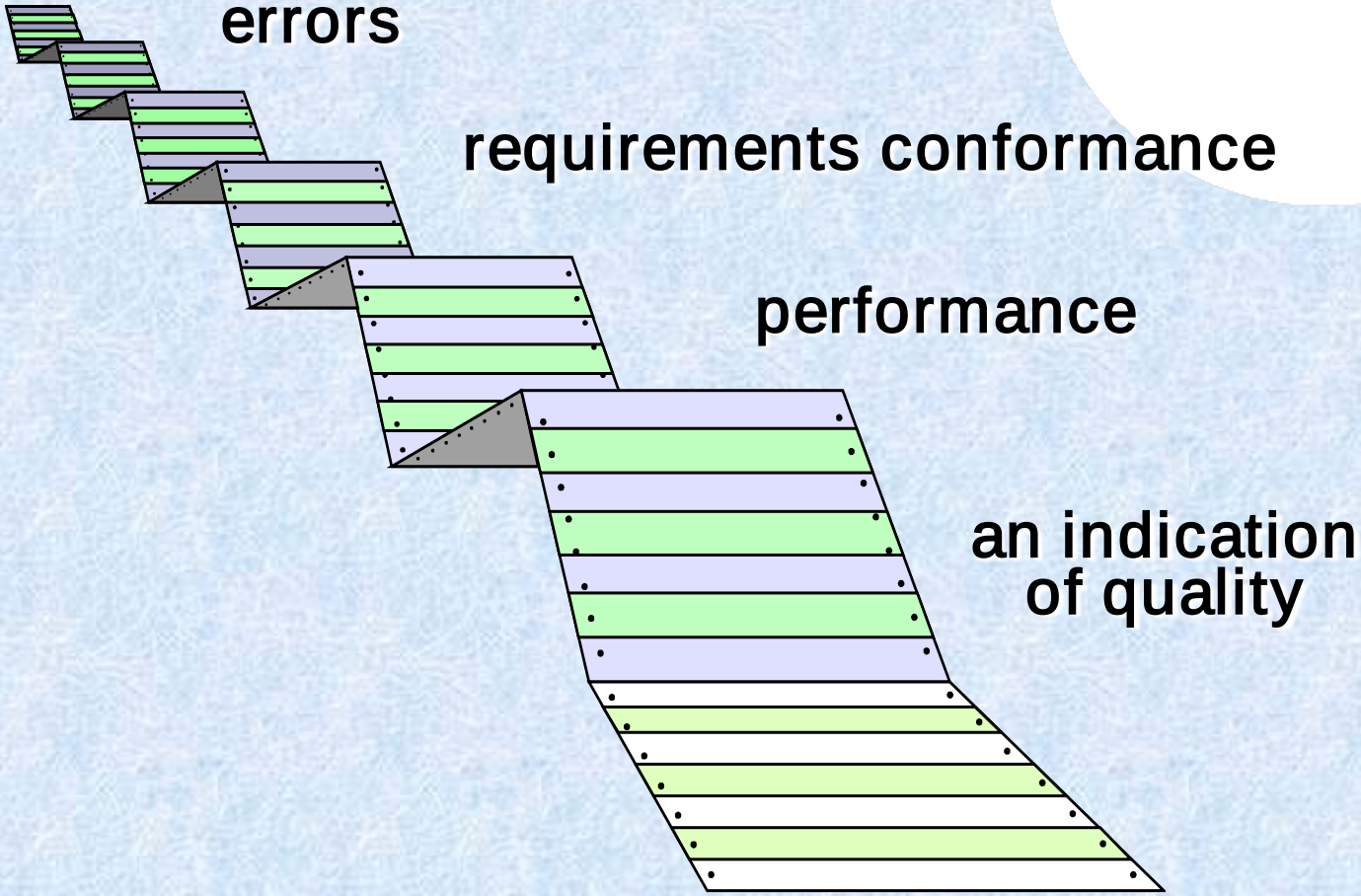
- In an Immature software organization, software processes are generally improvised by practitioners and their managers during the course of the project.
- Even if a software process has been specified, it is not rigorously followed or enforced.
- The immature software organization is reactive i.e. fire fighting
- A mature software organization, in contrast, possesses an organization wide ability for managing software development and maintenance processes.

- It accurately communicates the software process to both existing staff and new employees.
- The processes mandated are documented, usable and consistent with the way the work actually gets done.
- The process definitions are updated when necessary.
- Improvements are developed through controlled pilot-tests/ cost benefit analysis.
- There are objective, quantitative basis for judging product quality and analyzing problems with the product and process.

Software Testing

Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

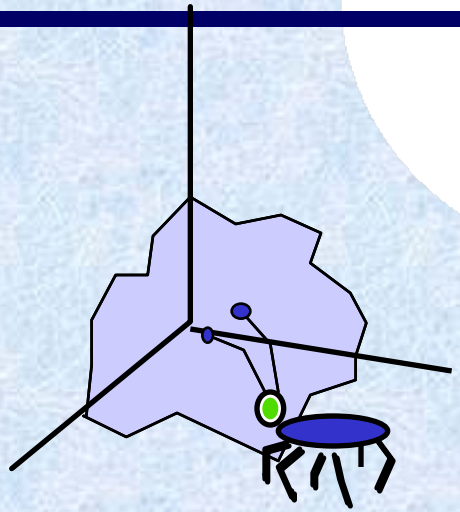
What Testing Shows



Test Case Design

"Bugs lurk in corners and congregate at boundaries ..."

Boris Beizer



OBJECTIVE to uncover errors

CRITERIA in a complete manner

CONSTRAINT with a minimum of effort and time

Who Tests the Software?

developer

Understands the system

but, will test "gently"

and, is driven by "delivery"

independent tester

Must learn about the system,

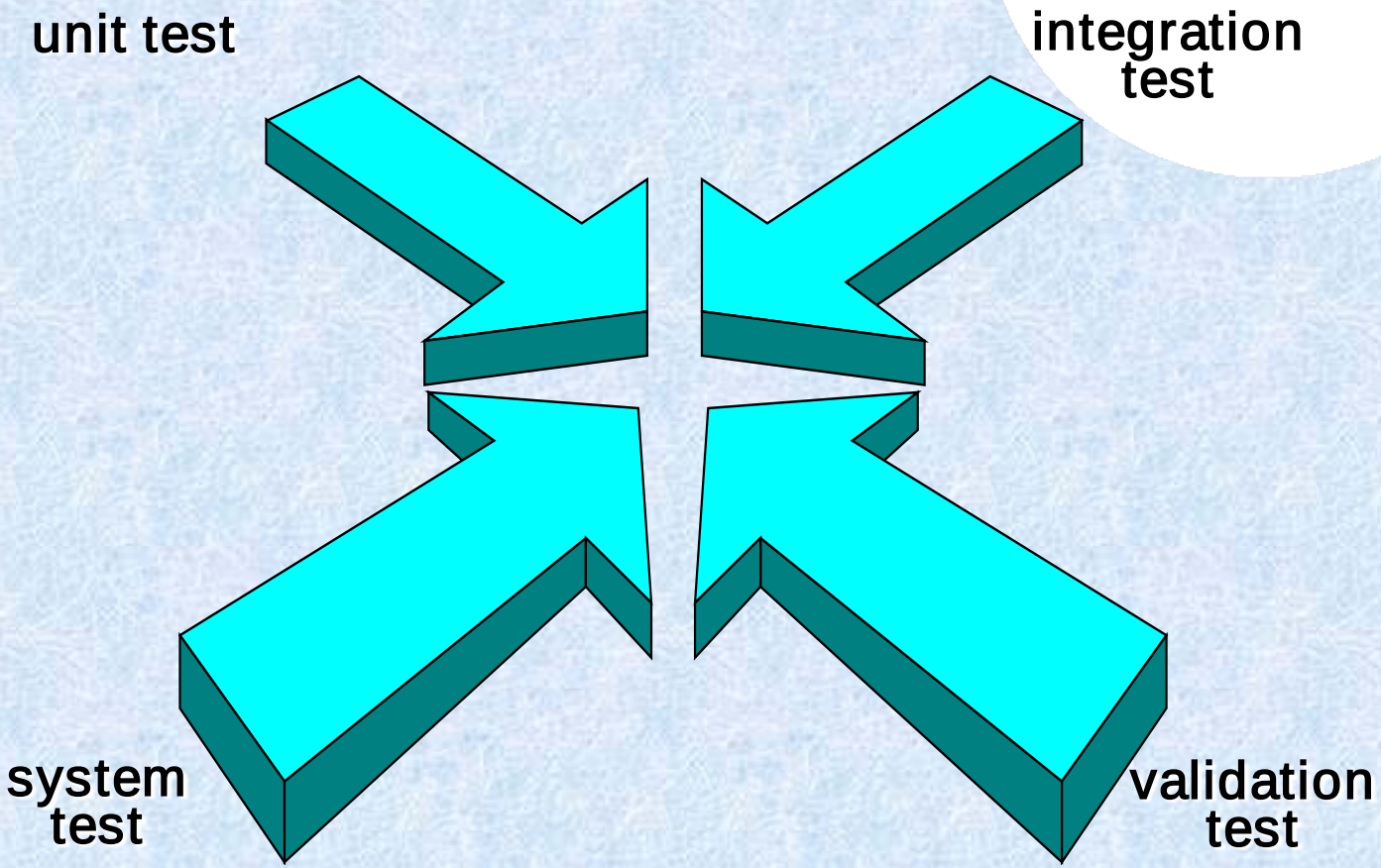
but, will attempt to break it

and, is driven by quality

Validation vs Verification

- Verification – Are we building the product right?
  - Is the code correct with respect to its specification?
- Validation – Are we building the right product?
  - Does the specification reflect what it should?

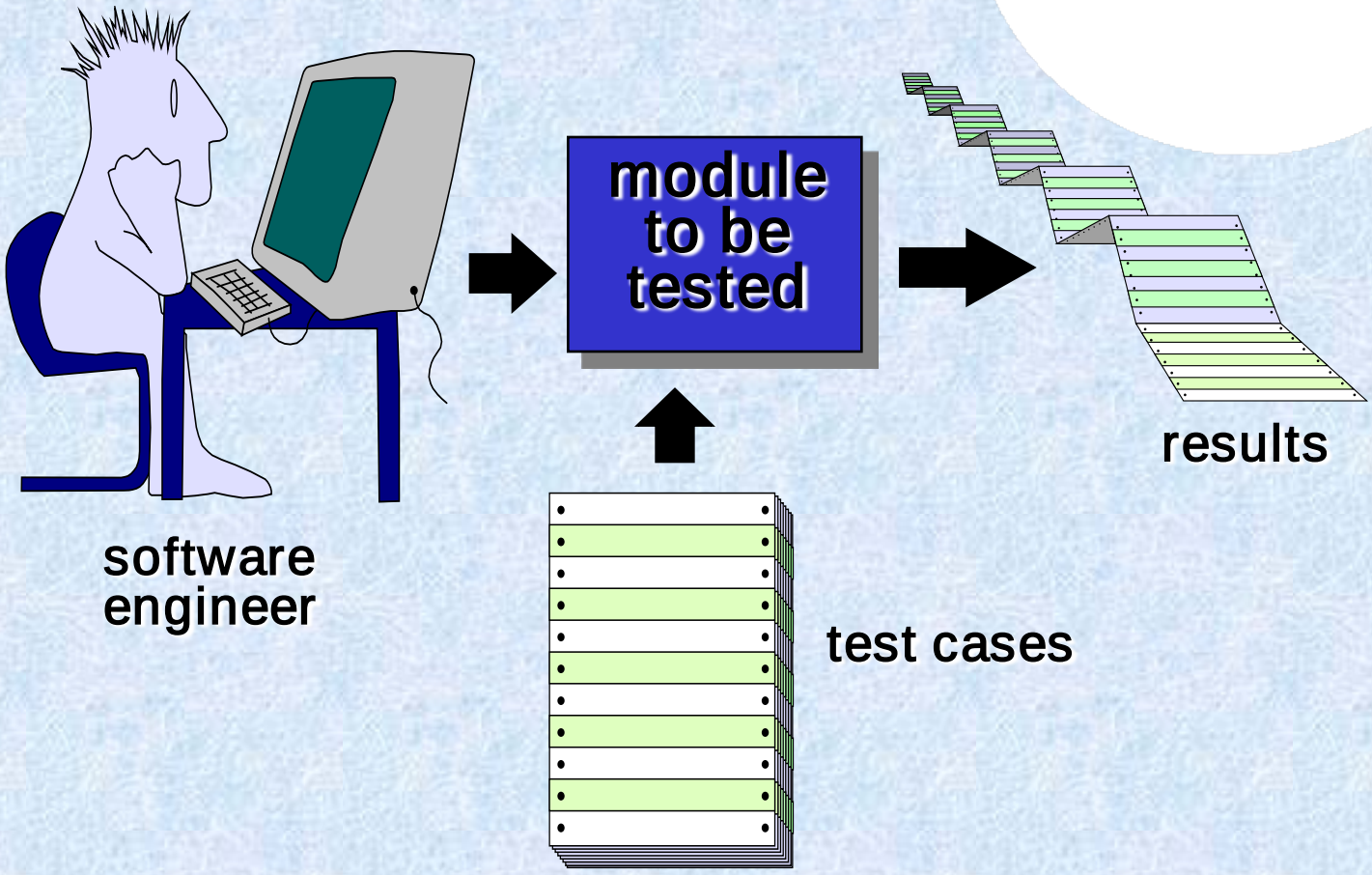
Testing Strategy



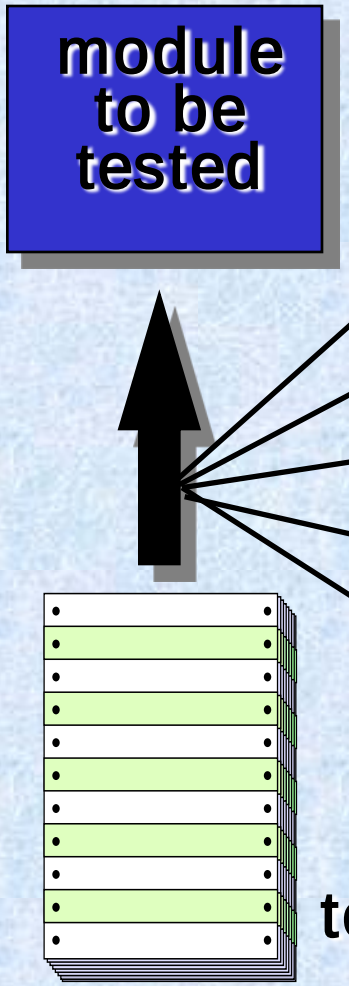
Testing Strategy

- Begin with unit testing and work your way up to system testing.
- Unit testing – test individual components (modules in procedural languages; classes in OO languages)
- Integration testing – test collections of components that must work together
- Validation testing – test the application as a whole against user requirements
- System testing – test the application in the context of an entire system

Unit Testing



Unit Testing



interface

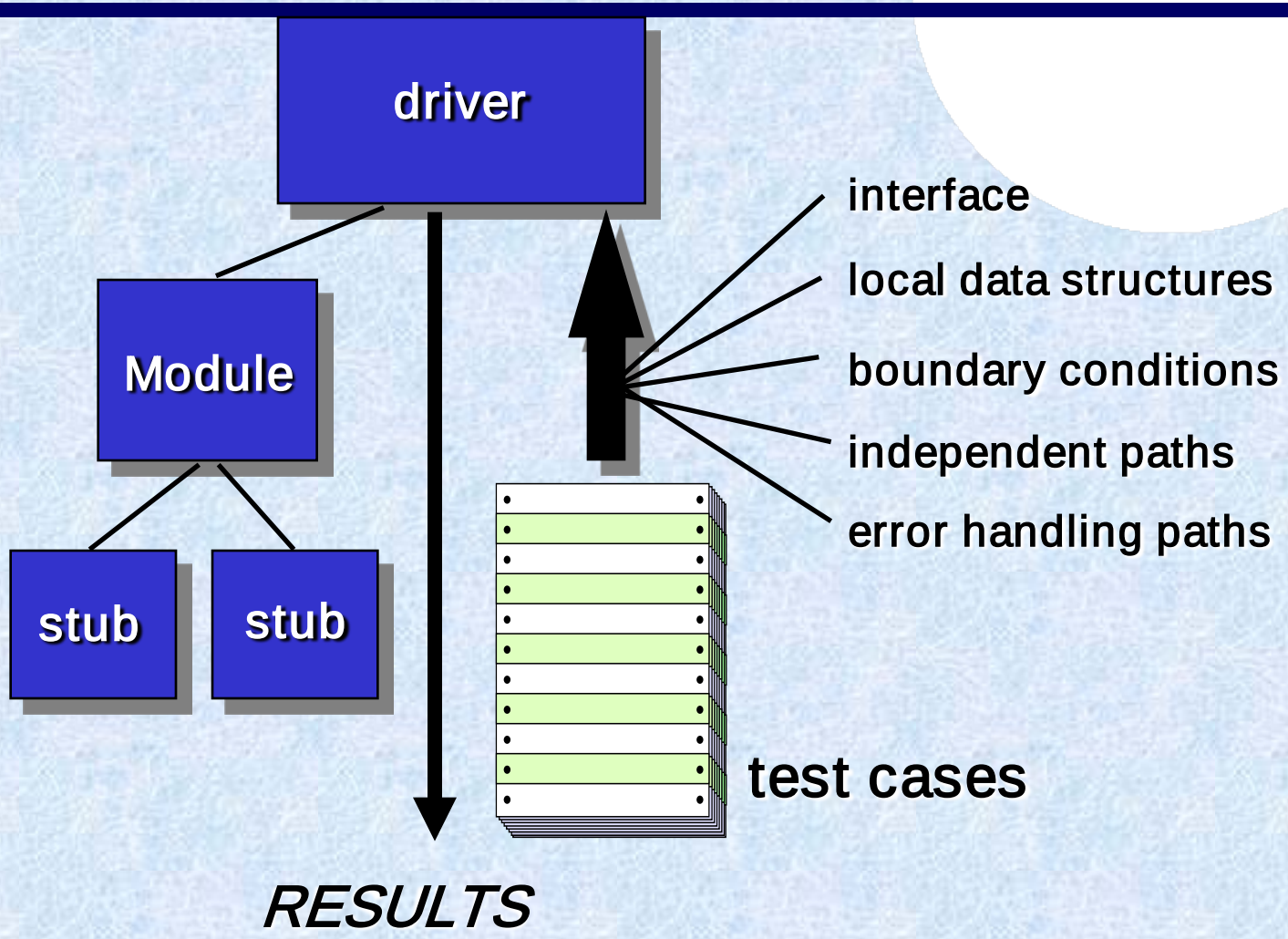
local data structures

boundary conditions

independent paths

error handling paths

Unit Test Environment

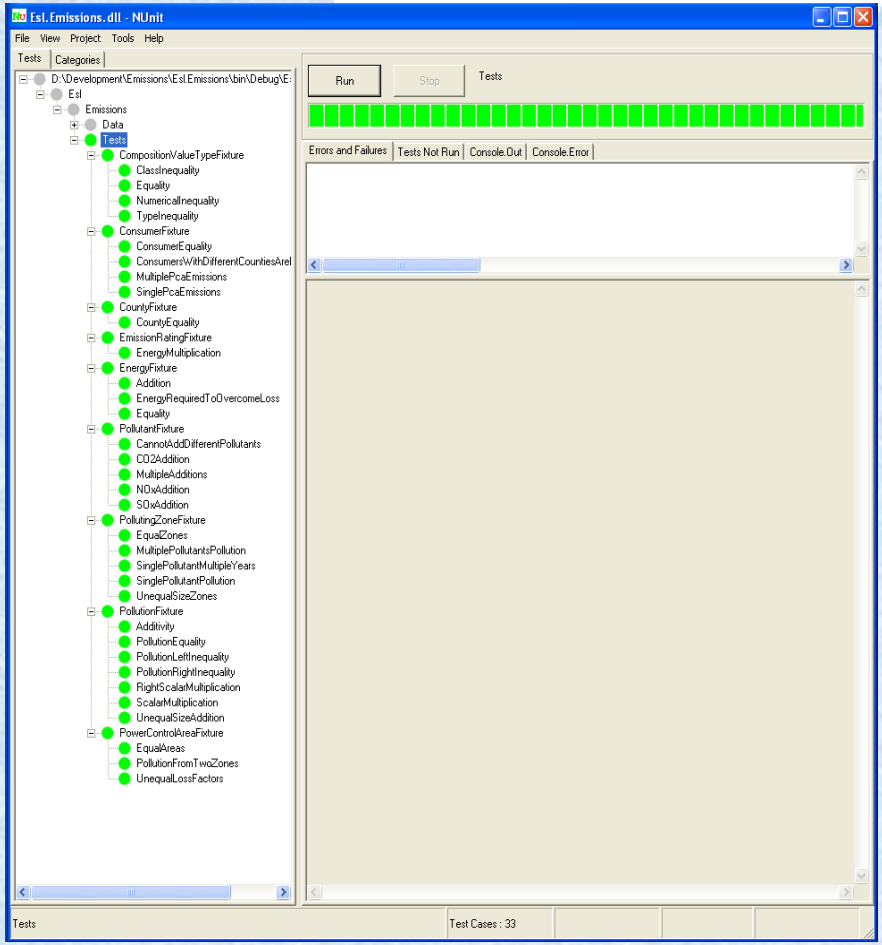


Unit Testing Details

- Interfaces tested for proper information flow.
- Local data are examined to ensure that integrity is maintained.
- Boundary conditions are tested.
- Basis path testing should be used.
- All error handling paths should be tested.
- Drivers and/or stubs need to be developed to test incomplete software.

Case Study: Unit Testing xUnit Tool:

- Unit testing tools in many languages
  - .NET: NUnit
  - Java: junit
  - C++: CppUnit

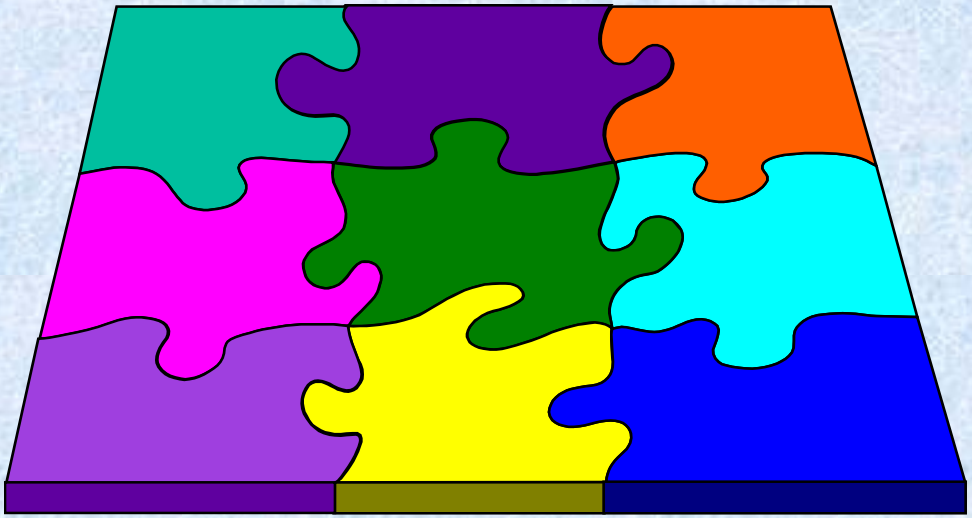


NUnit

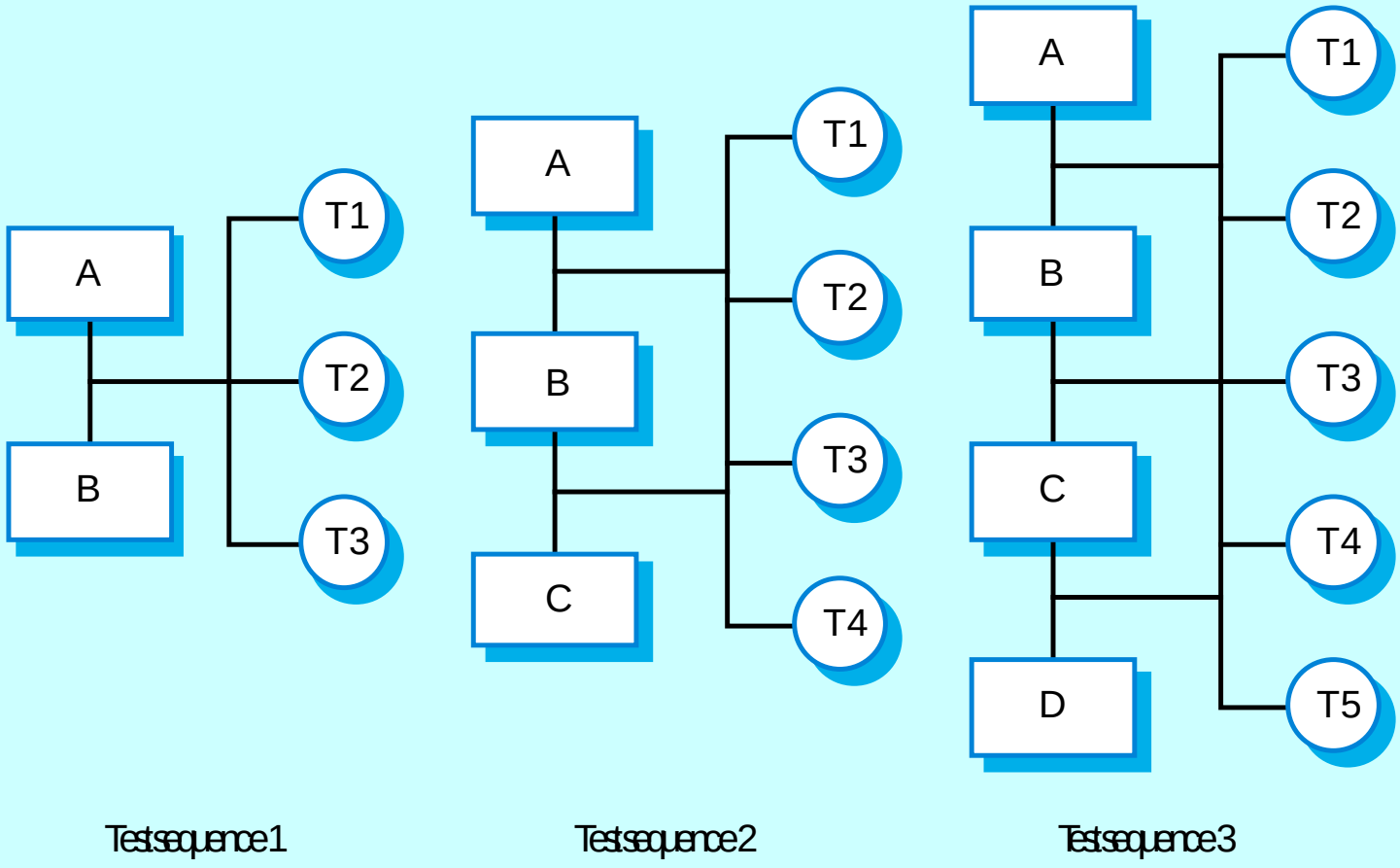
Integration Testing Strategies

Options:

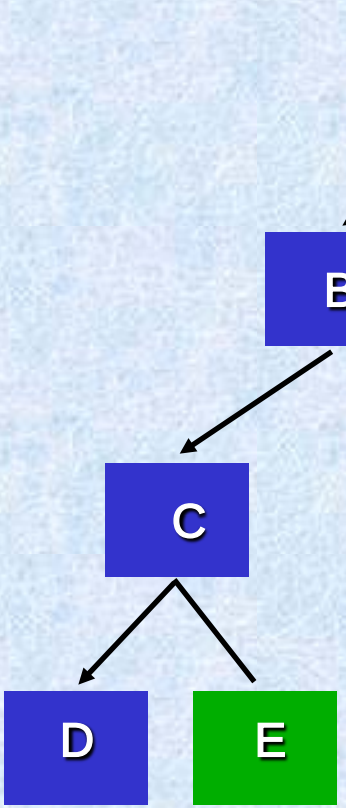
- the “big bang” approach
- an incremental construction strategy



Incremental integration testing



Top Down Integration

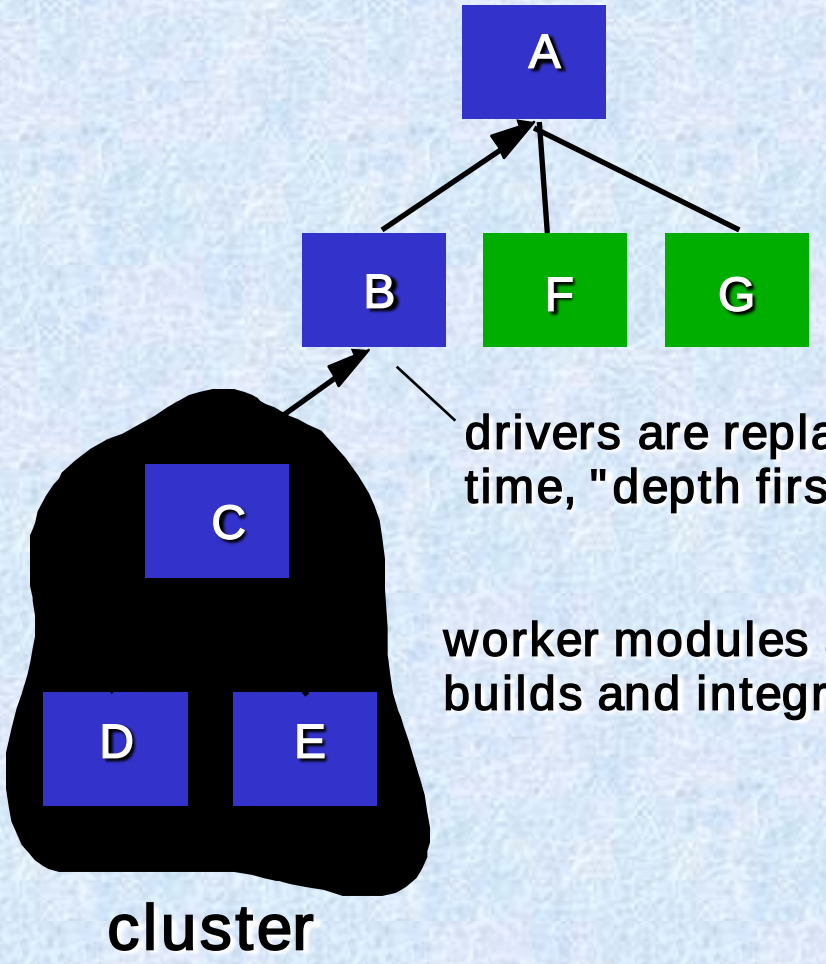


top module is tested with stubs

stubs are replaced one at a time, "depth first"

as new modules are integrated, some subset of tests is re-run

Bottom-Up Integration



Big-bang and Sandwich

Big-bang Approach

- First all the modules are individually tested
- Next all those modules are put together to construct the entire system which is tested as a whole

Sandwich Approach

- In this approach a system is integrated using a mix of top-down, bottom-up, and big-bang approaches
- A hierarchical system is viewed as consisting of three layers

45

- The bottom-up approach is applied to integrate the modules in the bottom-layer
- The top layer modules are integrated by using top-down approach
- The middle layer is integrated by using the big-bang approach after the top and the bottom layers have been integrated
- What is a “Good” Test? A good test is one that has a high probability of finding an error.

Validation Testing

- Ensure that each function or performance characteristic conforms to its specification.
- Deviations (deficiencies) must be negotiated with the customer to establish a means for resolving the errors.
- Configuration review or audit is used to ensure that all elements of the software configuration have been properly developed, cataloged, and documented to allow its support during its maintenance phase.

Regression Testing

- Check for defects propagated to other modules by changes made to existing program
  - To Reduce Side Effects
  - Representative sample of existing test cases is used to exercise all software functions.
  - Additional test cases focusing software functions likely to be affected by the change.
  - Tests cases that focus on the changed software components.

System Testing

- Involves integrating components to create a system or sub-system.
- May involve testing an increment to be delivered to the customer.
- Two phases:
  - Integration testing - the test team have access to the system source code. The system is tested as components are integrated.
  - Release testing - the test team test the complete system to be delivered as a black-box.

System Testing

- Recovery testing
  - checks system's ability to recover from failures
- Security testing
  - verifies that system protection mechanism prevents improper penetration or data alteration
- Stress testing
  - program is checked to see how well it deals with abnormal resource demands
- Performance testing
  - tests the run-time performance of software

Stress Testing

- Exercises the system beyond its maximum design load. Stressing the system often causes defects to come to light.
- Stressing the system test failure behaviour.. Systems should not fail catastrophically. Stress testing checks for unacceptable loss of service or data.
- Stress testing is particularly relevant to distributed systems that can exhibit severe degradation as a network becomes overloaded.

Performance Testing

- Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable.

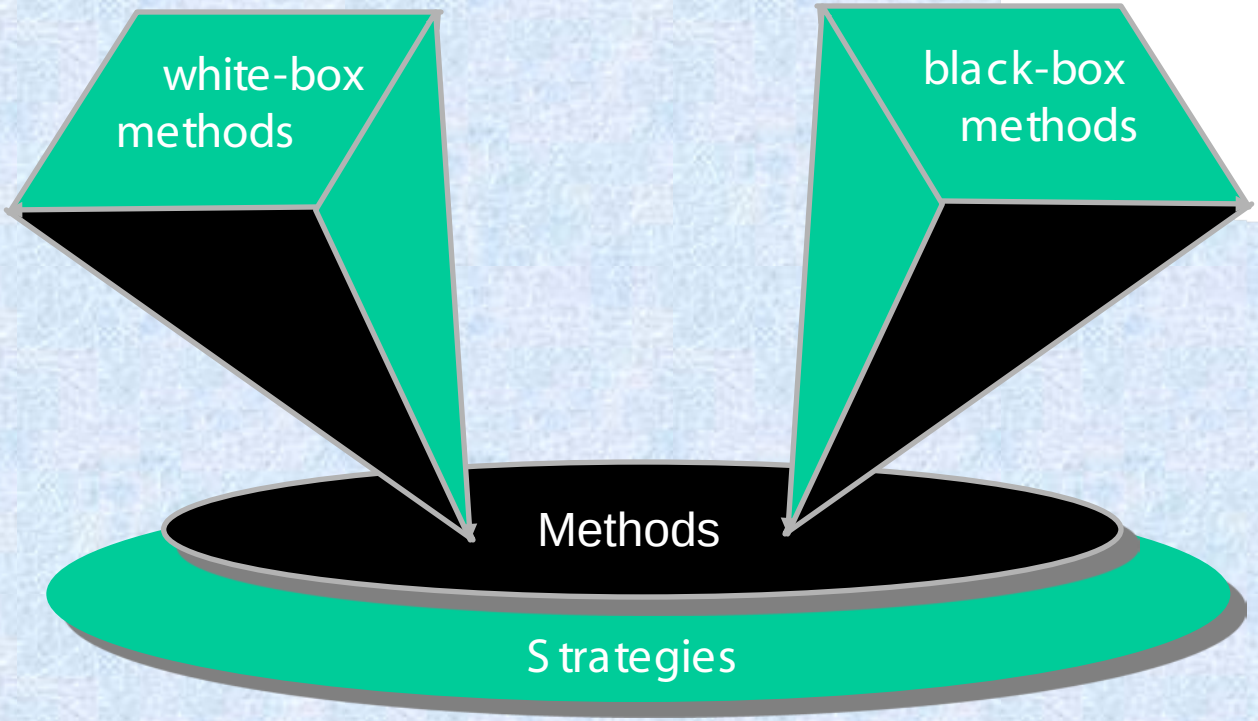
Performance Testing

- Stress test.
- Volume test.
- Configuration test (hardware & software).
- Compatibility.
- Regression tests.
- Security tests.
- Timing tests.
- Environmental tests.
- Quality tests.
- Recovery tests.
- Maintenance tests.
- Documentation tests.
- Human factors tests.

Performance Testing Tools

- Apache JMeter
- NeoLoad
- LoadRunner
- LoadUI
- WebLOAD
- WAPT
- Loadster
- LoadImpact
- Rational Performance Tester
- Testing Anywhere
- OpenSTA
- QEngine (ManageEngine)
- Loadstorm
- CloudTest
- Httperf

Software Testing Techniques



Test Case Design

- Two general software testing approaches:

Black-Box Testing:

- knowing the specific functions of a software,
- design tests to demonstrate each function and check its errors.
- Major focus:
- functions, operations, external interfaces, external data and information

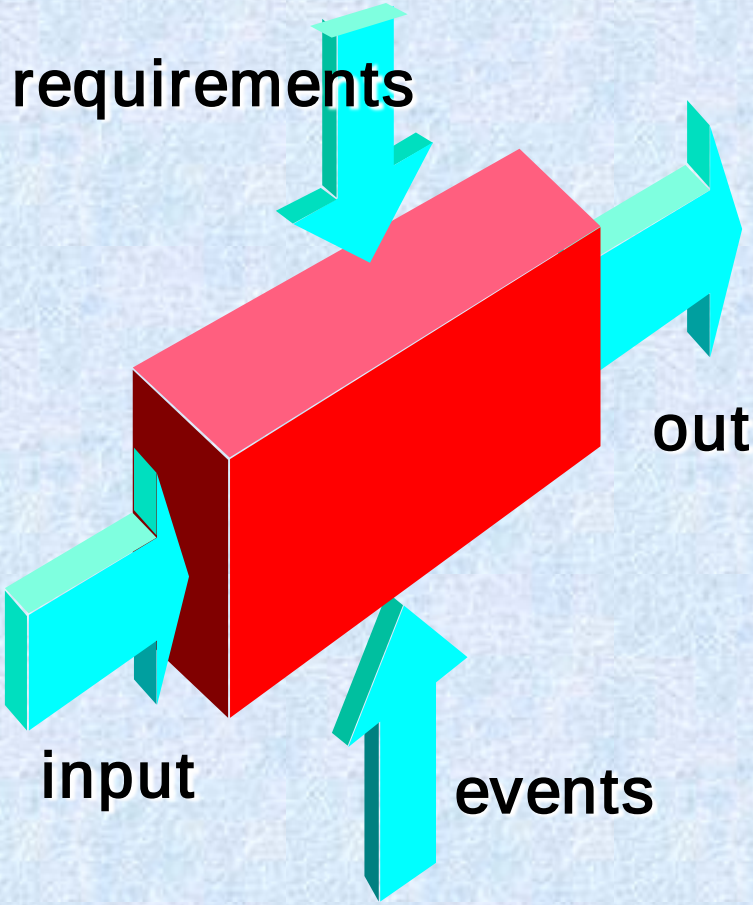
White-Box Testing:

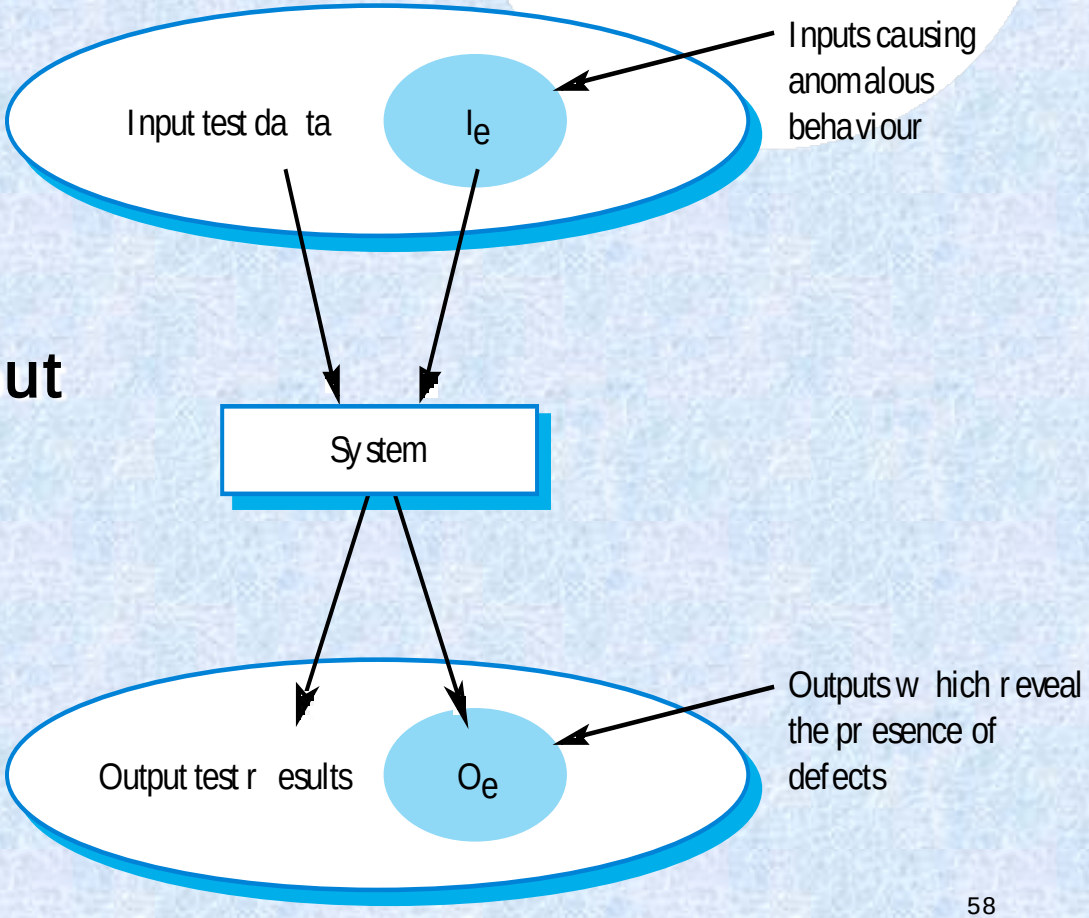
- knowing the internals of a software,
- design tests to exercise all internals of a software to make sure they operates according to specifications and designs
- Major focus: internal structures, logic paths, control flows, data flows internal data structures, conditions, loops, etc.

Black-Box (Behavioral) Testing

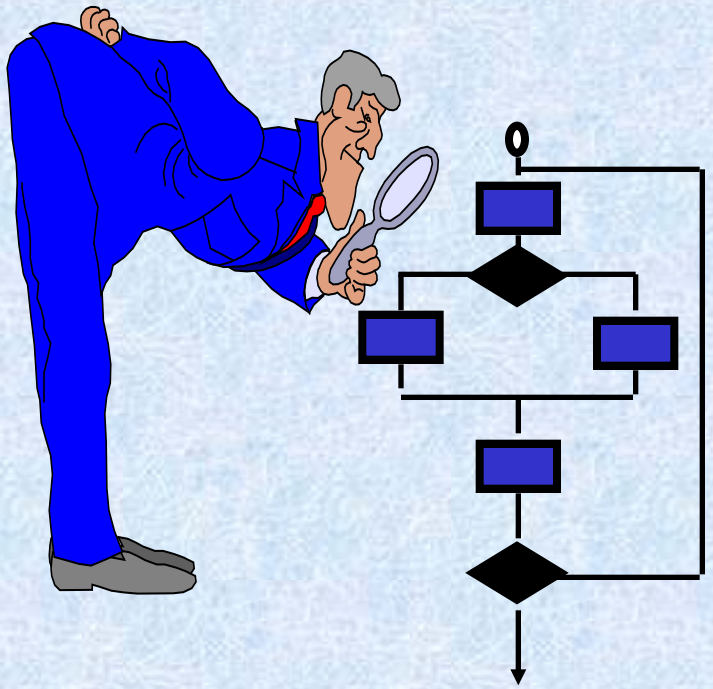
- Focuses on the functional requirements
- Tests are conducted at the SW Interface
- Examines Fundamental aspects of the system with little regard for internal logical structure
- Used to demonstrate that software functions are operational, that input is properly accepted and output is correctly produced, and that the integrity of external information (e.g., a database) is maintained.
- It attempts to find errors in 1.Incorrect or missing function 2. interface error 3. error in DS or external DB access 4. Behavior or performance error etc.

Black-Box Testing





White-Box Testing



... our goal is to ensure that all statements and conditions have been executed at least once ...

- Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh," that is, internal operations are performed according to specifications and all internal components have been adequately exercised.
- It is predicated on close examination of procedural detail. Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and/or loops.
- Objective is to exercise all program statements (not all path combinations).
- This tests can be designed only after component level design (Source Code (logical detail)).

- White-box testing, sometimes called *glass-box testing,* is a test case design method that uses the control structure of the procedural design to derive test cases.
- Using white-box testing methods, the software engineer can derive test cases that :

(1) Guarantee that all independent paths within a module have been exercised at least once.

(2) Exercise all logical decisions on their true and false sides.

(3) Execute all loops at their boundaries and within their operational bounds.

(4) Exercise internal data structures to ensure their validity.

Acceptance Testing

- Making sure the software works correctly for intended user in his or her normal work environment.
- Alpha test
  - version of the complete software is tested by customer under the supervision of the developer at the developer's site (Controlled Environment)
- Beta test
  - version of the complete software is tested by customer at his or her own site without the developer being present (Live Environment)

Acceptance Testing Approaches

- **Benchmark test:** A benchmark is a point of reference by which something can be measured.

  - Compares the performance of new or unknown target-of-test to a known reference standard, such as existing software or measurements

- **Pilot testing:** Testing that involves the users just before actual release to ensure that users become familiar with the release contents and ultimately accept it.

  - Pilot testing involves having a group of end users try the system prior to its full deployment in order to give feedback

- **Parallel testing.**

Debugging

- Debugging (removal of a defect) occurs as a consequence of successful testing.
- Some people better at debugging than others.
- Is the cause of the bug reproduced in another part of the program?
- What “next bug” might be introduced by the fix that is being proposed?
- What could have been done to prevent this bug in the first place?

Debugging Approaches

- Brute force
  - Memory dumps and run-time traces are examined for clues to error causes
- Backtracking
  - Source code is examined by looking backwards from symptom to potential causes of errors
- Cause elimination
  - Uses binary partitioning to reduce the number of locations potential where errors can exist

Summary

- Testing is an integral part of the system development function, not an afterthought:
  - Testing starts with the requirements not the code
  - Testing is a static and dynamic activity
  - Prevention is better than cure
  - The sooner you find the fault the cheaper it is to fix
  - Create re-usable ‘testware’
  - Process first then tools
  - Not everyone can test well, use professional testers
  - Planned testing in a controlled environment provides objective metrics

  - To gain a Return on Investment you must first Invest

Thank You