Lecture X:

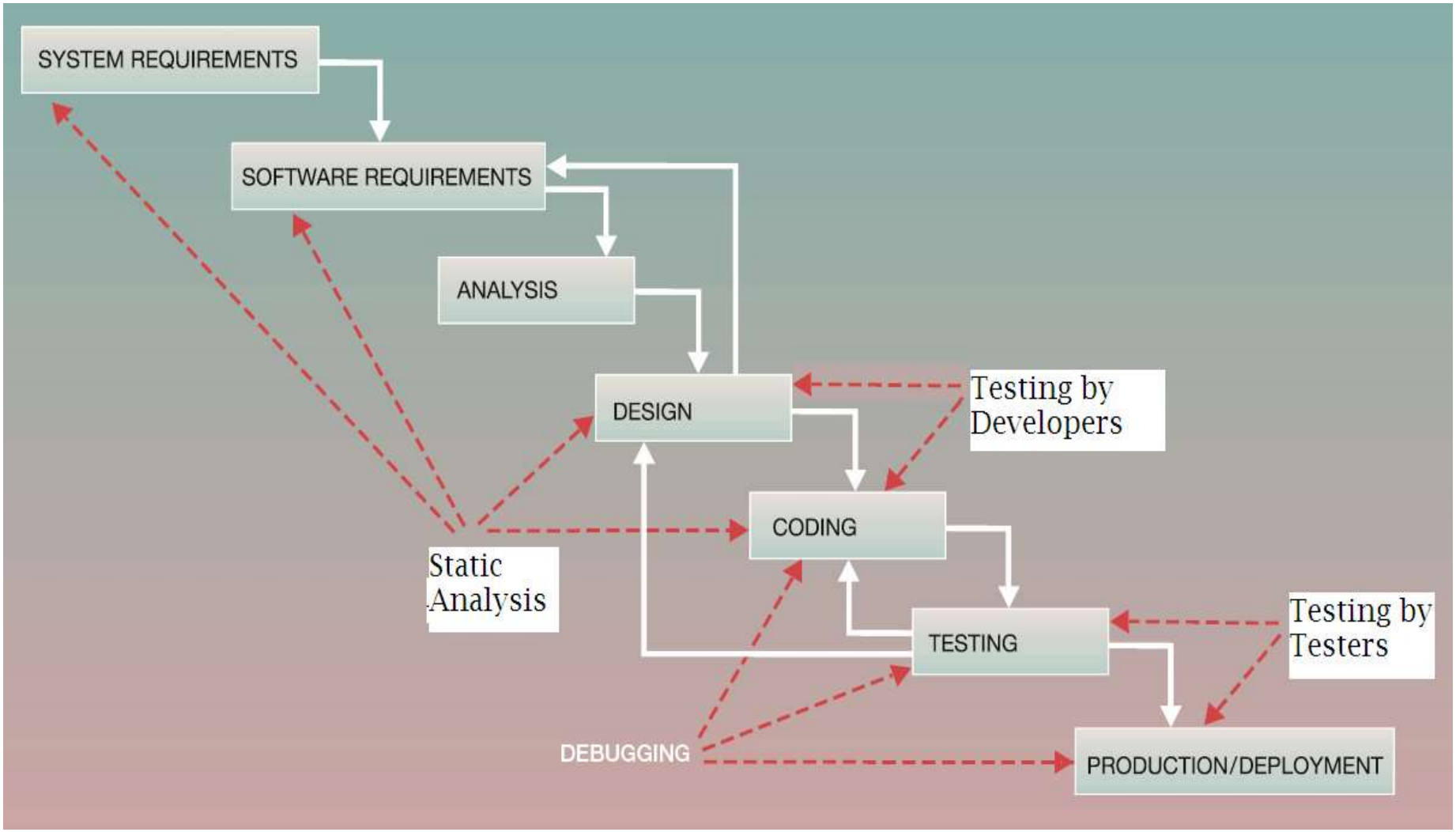
Software Testing

General Views

Topics to be covered

- Software Testing Principles
- Types of Software Tests
- Test Planning

What is Software testing?



What is Software testing?

- This is the **execution of a program to find its faults**.

- While **more time** typically is **spent on testing** **<u>than</u>** in any other phase of software development, there is **<u>considerable</u>** **<u>confusion</u>** about its purpose.

- Many software professionals, for example, believe that **tests are run to show that the program works** rather than to learn about its faults.

What is Software testing? (2)

Glenford Myers(2012) has provided some useful testing definitions:

- **Testing**

The process of **executing a program** (or part of a program) with the **intention** of **finding errors**.

- **Verification**

An **attempt to find errors** by executing a program in **a test or simulated environment** (it is now preferable to view verification as the process of proving the program's correctness)

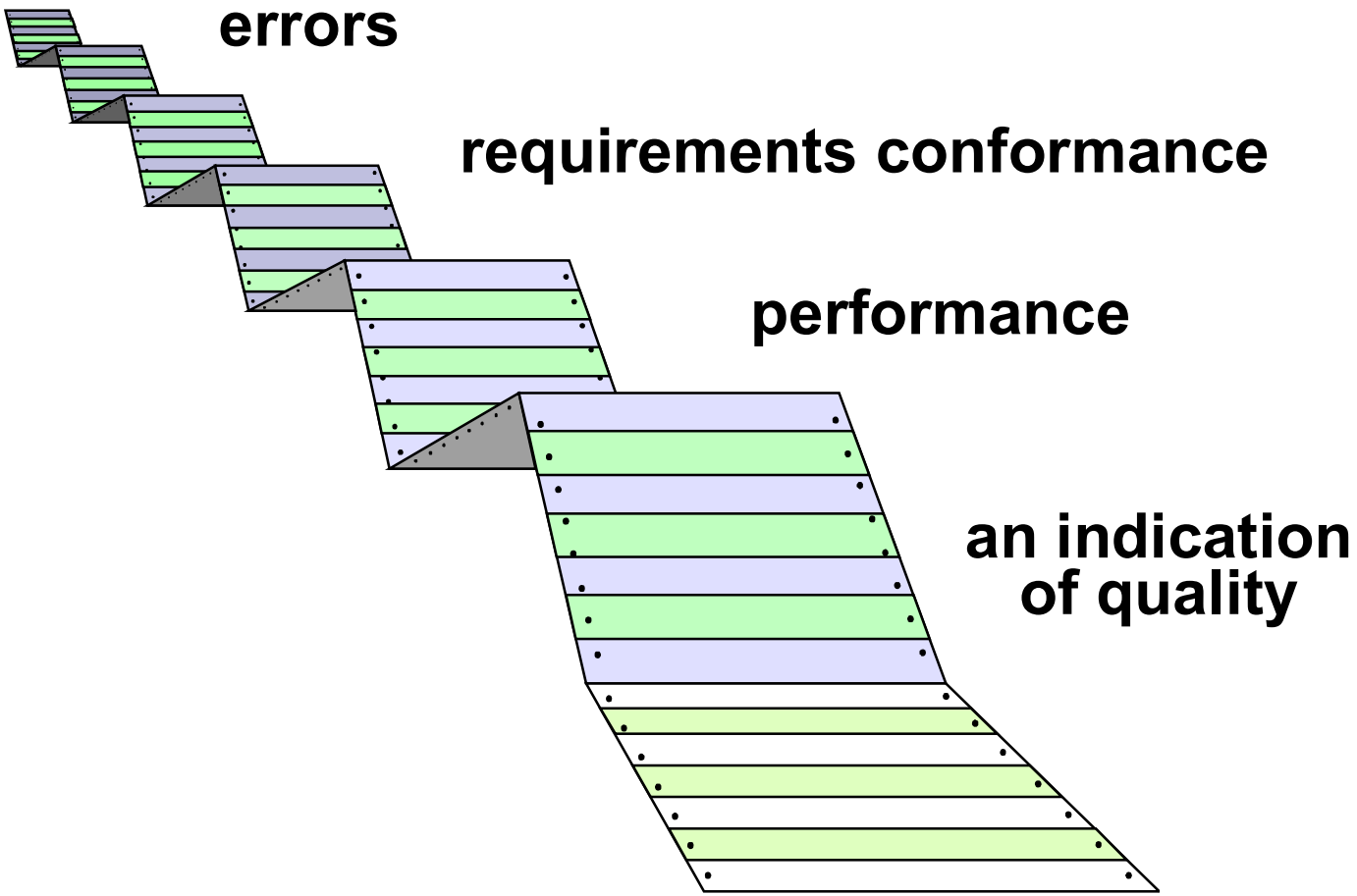
- **Validation**

An **attempt to find errors** by executing a program in **a real environment**.

- **Debugging**

Diagnosing the precise nature of a known error and then **correcting it** (debugging is a correction and not a testing activity)

What Software testing shows



Who tests software?

developer

independent tester

Understands the system but, will test "gently" and, is driven by "delivery"

Must learn about the system, but, will attempt to break it and, is driven by quality

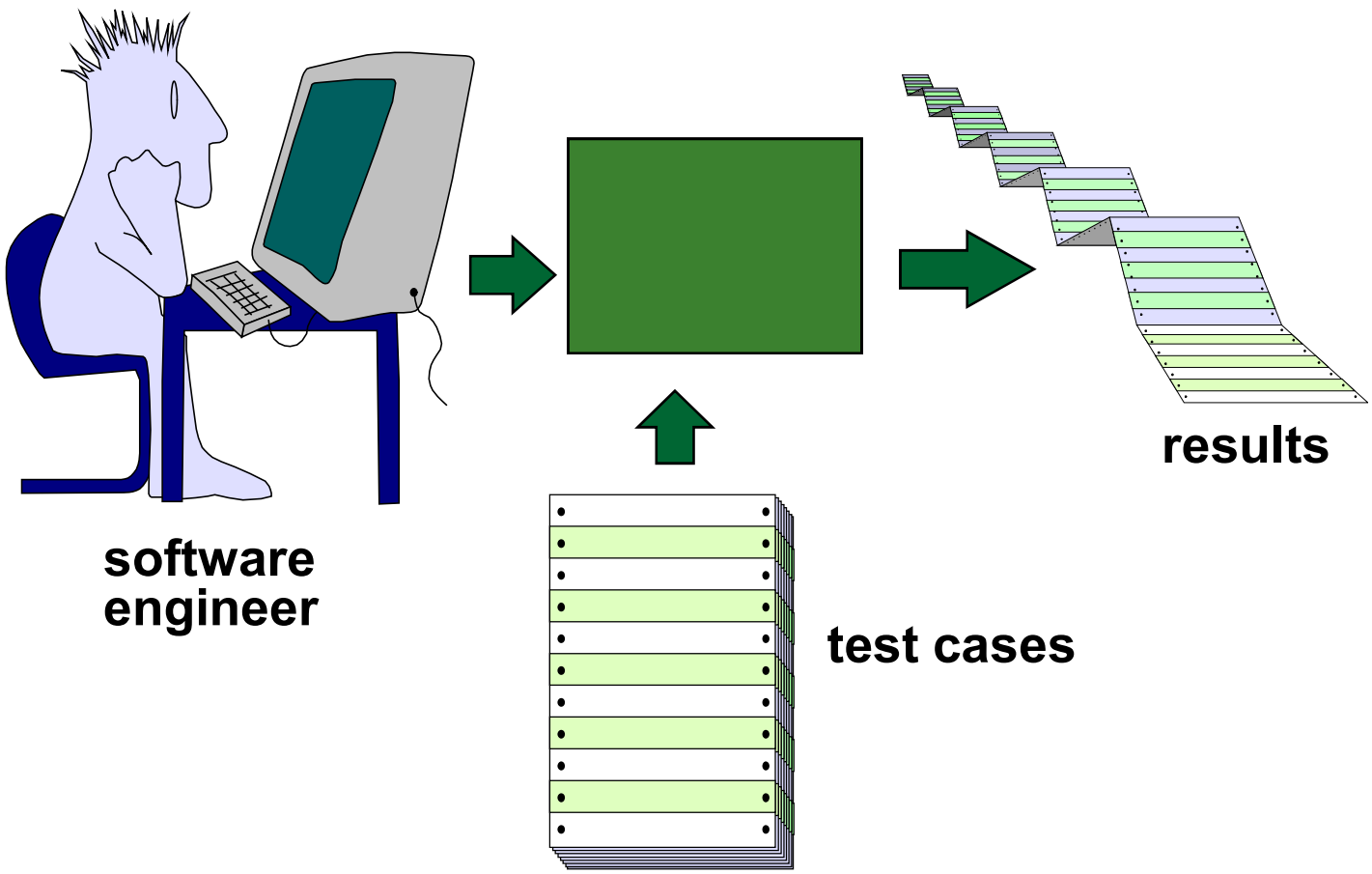
Types of Software Testing

Unit Testing

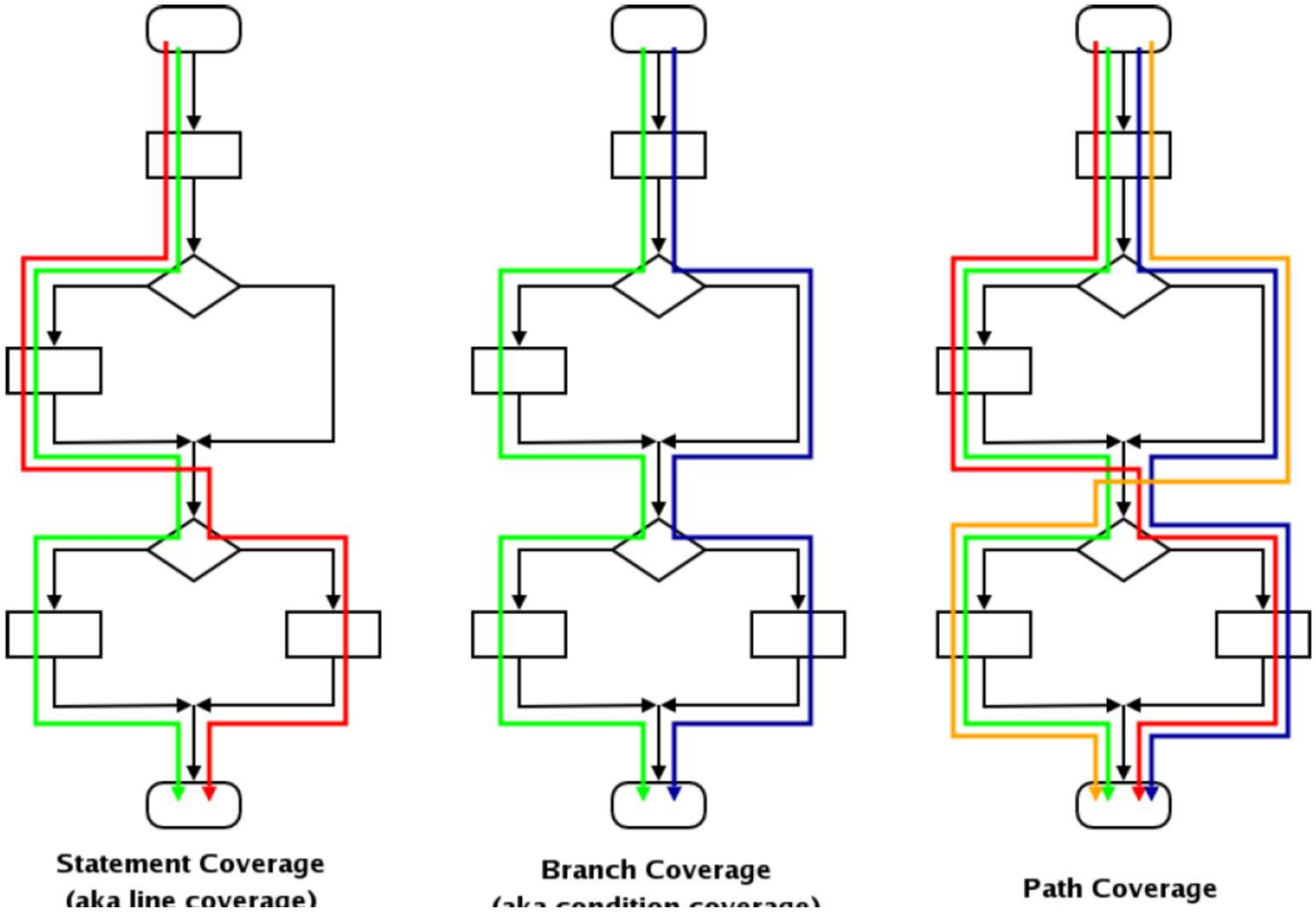
Unit Testing (White Box)

- Individual components are tested
- It is a path test
- To focus on a relatively small segment of code and aim to exercise a high percentage of the internal path
- **Disadvantage**: the tester may be biased by previous experience. And the test value may not cover all possible values

Unit Testing (White Box)



Unit Testing (White Box)



Integration Testing

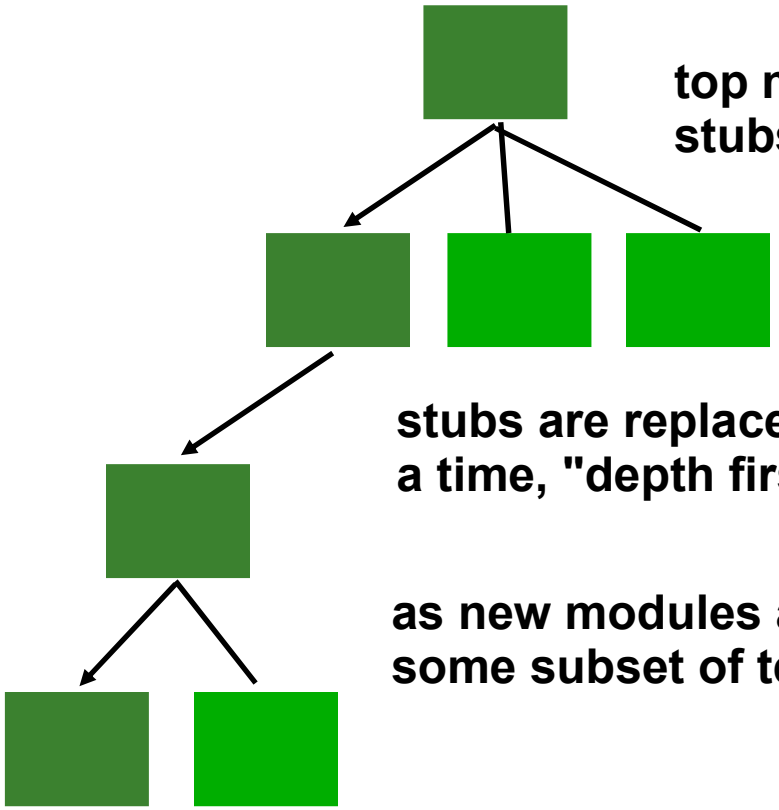
Integration Testing

- Top-down Integration Test
- Bottom-up Integration Test

Top-down Integration Test

- The control program is tested first
- Modules are integrated one at a time
- Emphasize on interface testing
- Advantages:
  - No test drivers ('calling modules') needed
  - Interface errors are discovered early
  - Modular features aid debugging
- Disadvantages:
  - Test stubs ('called modules') are needed
  - Errors in critical modules at low levels are found late

Top-down Testing



top module is tested with stubs

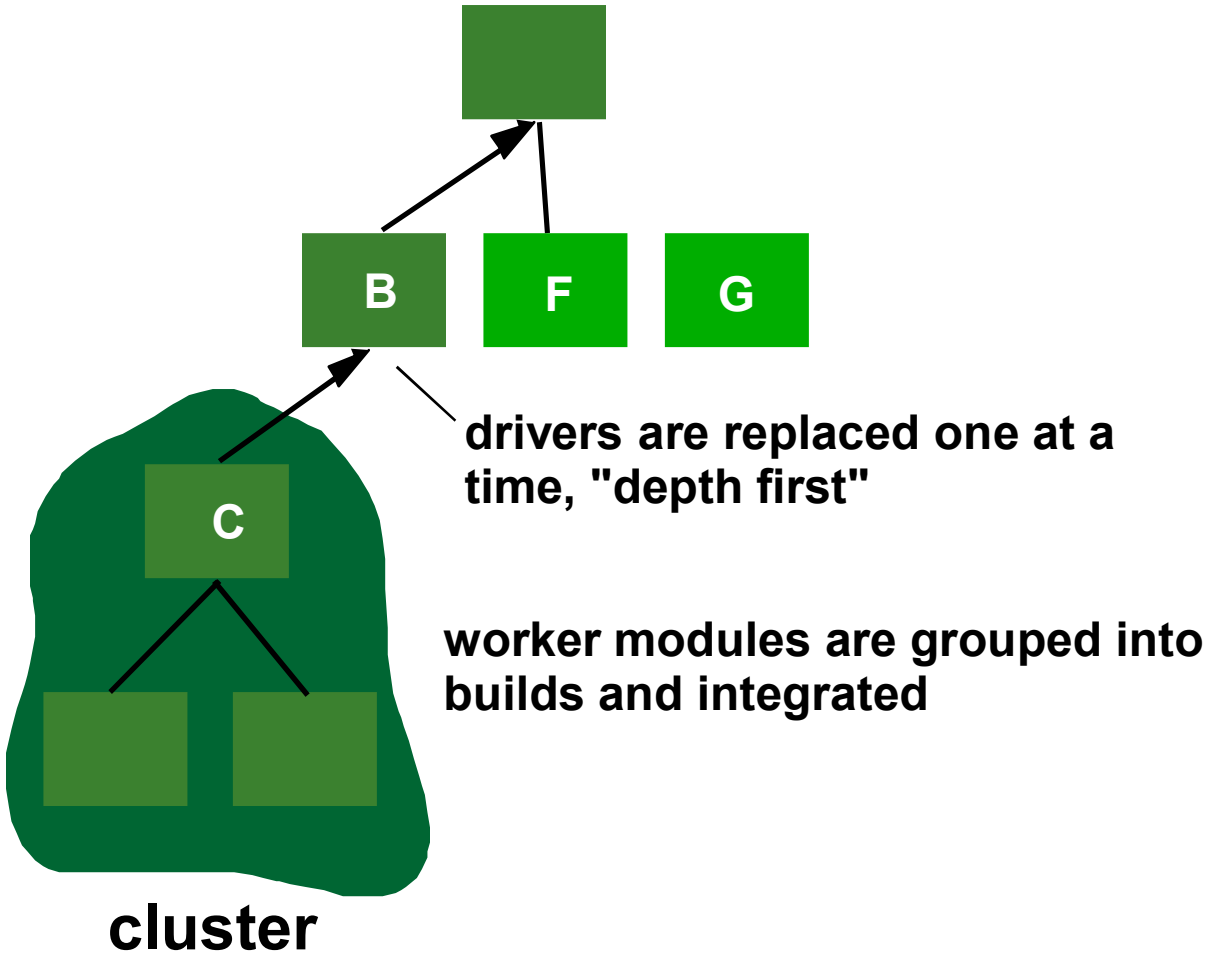
stubs are replaced one at a time, "depth first"

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

Bottom-up Integration Test

- Allow early testing aimed at proving feasibility.
- Emphasize on module functionality and performance
- Advantages:
  - No test stubs ('called modules') are needed
  - Errors in critical modules are found early
- Disadvantages:
  - Test drivers ('calling programs') are needed
  - Interface errors are discovered late

Bottom-up testing



Function Testing

Function Testing (Black Box)

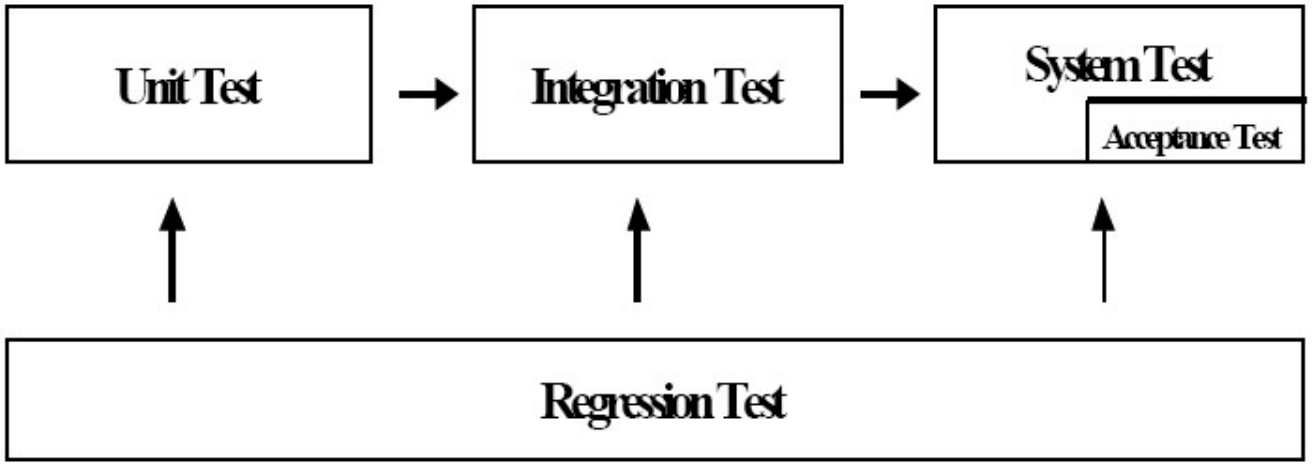
- Designed to exercise the interface to its external specifications
- Testers not biased by knowledge of the program's design
- Disadvantages:
  - The need for explicitly stated requirements
  - Only cover a small portion of the possible test conditions, *when errors are found – a closer examination is required*

Regression Testing

Regression Testing

- Test the effects of the newly introduced changes on all the previously integrated code
- The common strategy is to accumulate a comprehensive regression bucket but also to define a subset
- The full bucket is run only occasionally, but the subset is run against every spin
- Disadvantages:
  - To decide how much of a subset to use and which tests to select

Regression Testing



Real-Time Testing

Real-Time Testing

- Real-Time testing is necessary because the deployment of a system is usually more complicated than development system

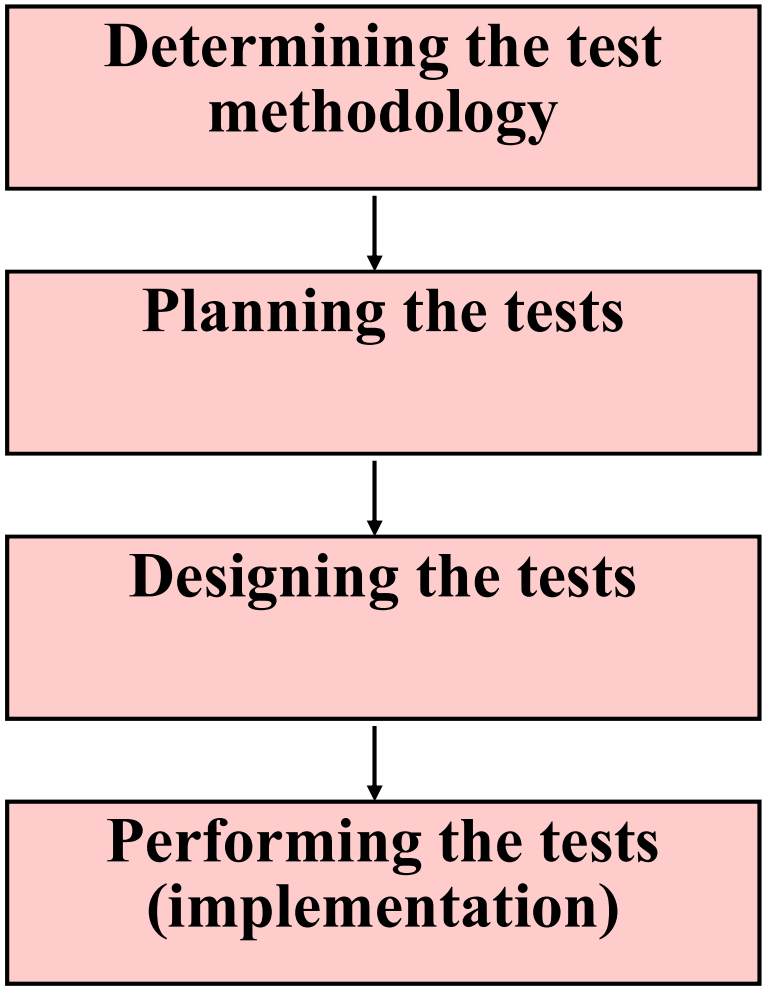
- Rules apply for testing real time system
  - Evaluate possible **deadlocks**, thrashing to special timing conditions
  - Use tests to **simulate hardware faults**
  - Use hardware simulation to **stress the software design**
  - Design ways to **simulate modules** missing in the development system

The Testing Process

The Testing Process

- Testing is **done throughout** the **development process**.
- Testing is **divided into phases** beginning in the design phase and ending at the customer's site.
- **Testing process** is illustrated in the next slide.
- The **two fundamental decisions** that must be made before planning for testing can occur are:
  - **What** is the **required software quality standard**, and
  - **What** is the **software testing strategy**.

The Testing Process (2)



Determining the Appropriate Software Quality Standard

- **Different standards** required for **different software applications**. e.g. safety-critical software or aircraft instrumentation - critical.

- In other cases, **a medium-level quality** would be **sufficient**, and

- So, the **expected damage** resulting from failed software **impacts** standard of software quality.

- Samples of damage to customers and users **as well as** to developers are shown on the next two slides:

Classification of Software Failure Damages

- Damages to Customers and Users
  - Endangers the **safety** of human beings
  - Affects an **essential organizational function with no system replacement capability available**
  - **Affects** functioning of **firmware**, causing malfunction of an entire system
  - Affects **proper functioning** of software packages for business applications
  - Affects **proper functioning** of software packages for a private customer
  - Affects **functioning** of a firmware application but without affecting the entire system.
  - **Inconveniences** the user but does not prevent accomplishment of the system's capabilities

Classification of Software Failure Damages (2)

- Damages to Software Developer
  - **Financial losses**
    - Damages paid for physical injuries
      - Aircraft or auto instrumentation; health equipment…. Law suites!!
    - Damages paid to organizations for malfunctioning of software
      - Companies have many lawyers on staff!!!
    - Purchase cost reimbursed to customers
    - High maintenance expenses for repair of failed systems
  - **Non-quantitative damages**
    - <u>Expected</u> to affect future sales
    - Substantially reduced current sales

Determining Software Testing Strategy

- **Big Bang or Incremental?** So, do we want the testing strategy to be big bang or incremental?
  - Major testing at end ***in the past***….
  - If incremental, top down or bottom up?
- Which parts of the testing plan should be done using **White Box testing**? **Black box**?
- Which parts of the test plan should be done using an **automated test model**?

Planning the Tests

- We need to undertake:
  - Unit tests
  - Integration tests, and
  - System Tests.

- **Unit tests** deal with **small hunks** – modules, functions, objects, classes;

- **Integration tests** deal with **units constituting** a **subsystem** or other major hunks of capability, and

- **System tests** refer to the **entire software package** or system.

- These are often done by different constituencies!!

Lots of Questions

- So we first need to consider five basic issues:
  - **What** to test
  - **Which sources** do we use for test cases
  - **Who** is to perform the tests
  - **Where** to perform the tests, and
  - **When** to terminate the tests.
- Questions with not so obvious answers!

What to Test

- We **would like** to test everything.
- **Not very practical**.

- **Cannot** undertake exhaustive testing…
  - Number of paths is infinite….

- **Consider**:
  - Do we totally test modules that are ***98% reused?***
  - Do we really need to ***test things*** that have been ***repeatedly tested*** with ***only slight changes***?
  - How about testing by ***newbies***?
  - Testing on ***sensitive modules*** that ***pose lots*** of ***risk***?

What to Test (2)

- So, which modules need to be **unit tested**?
- Which **integrations** should be tested?
- Maybe low priority applications tested in **unit testing** may not be **needed** or **included** in the **system tests**….

- **Lots of planning** is needed, as testing **IS** a ***very expensive undertaking!***

Rating Units, Integrations, and Applications

- We need to **rate** these **issues** to determine their priority in the testing plan.

- Rate based on two factors:
  - **Damage severity level** – severity of results if module / application fails.
    - How much **damage** is done??
    - Will it destroy our business? Our reputation??
  - **Software risk level** – what is the **probability** of failure.

Which Sources Should be Used for Test Cases?

- Do we use **live test cases** or **synthetic test cases**.
- All three types of tests should consider these.

- Use **live data** or **contrived** (dummy) **data**??
- What do you think??

- Also need to consider **single** / **combined tests** and the number of tests.

- How about if the testing is top down? Bottom up? What sources do you think might be needed then??

Who Performs the Tests?

- **Unit Testing** – done by the *programmer* and/or *development team*.
- **Integration Testing** – can be the *development team* or a *testing unit*.
- **System Testing** – usually done by an *independent testing team* (internal or external (consultants) team.
- For small companies, another testing **team** from another development team can be used and **swapped**.
- Can always **outsource** testing too.

Where to Perform the Tests?

- Typically at the **software developer's** site.
- For system tests, test at **developer's** or **customer's** site (target site).
- If outsourced, testing can be done at **consultant's** site.

When are Tests Terminated?

- This is always the **$64,000 question**!!!
- Decision normally applies to **system tests**.
- Five typical alternatives:

1. Completed Implementation Route

- Test until all is error free. (good luck)
- All testing, regression testing;
- Disregards budget and timetable constraints.
- Applies 'To-Perfection Approach'

When are Tests Terminated? (3)

2. Mathematical Models Application Route:

- Here modeling is used to **estimate percentage of undetected errors** based on rate of error detection.
- When **detection rate falls below** a certain level, **stop**.
- **Disadvantage**: math model may not fully represent the project's characteristics.
  - Thus testing may be cut short or extended too far.
- **Advantage:** Well-defined stopping point.

When are Tests Terminated? (4)

3. Error Seeding Route

- Here, we seed errors prior to testing.

- Underlying assumption is that percentage of **discovered** seeded errors will correspond to the percentage of **real** errors detected.
- Stop once **residual percentage** of undetected seeded errors reaches a predefined level considered acceptable for 'passing' the system.

- **Disadvantages**: **extra workload** for testers; also based on past experiences of some testers;

- Seeding method **can not accurately estimate** the residual rate of undetected errors in unfamiliar systems.

When are Tests Terminated? (5)

4. The dual independent testing teams route:

- Here **two teams implement** the testing process **independently**.
  - Compare lists of detected errors.
  - Calculate the number of errors left undetected
  - Lots of statistics here.

- **High costs**. Justified when??

When are Tests Terminated? (6)

5. Termination after resources have petered out.

- This means **stop gradually** when budgets or time for **testing has run out**.
- **Very common** in industry

Test Planning

What is Test Planning?

- Define the functions, roles and methods for all test phases
- Test planning usually start during the requirements phase
- Major test plan elements are:
  - Objectives for each test phase
  - Schedules and responsibilities for each test activity
  - Availability of tools, facilities and test libraries
  - Set the criteria for test completion

Test Design and Software Test Plan (STP)

- **Products of Test Design**
  - Detailed **design** and **procedures** for **each test**
  - The **input database** / **files** for **testing**.

- There are standard **software test plans** (STP) templates

Software Test Description (STD) Template

1. Scope of the tests

1.1 The software package to be tested (name, version and revision)

1.2 The documents providing the basis for the designed tests (name and version for each document)

2. Test environment (for each test)

2.1 Test identification (the test details are documented in the STP)

2.2 Detailed description of the operating system and hardware configuration and the required switch settings for the tests

2.3 Instructions for software loading

Software Test Description (STD) Template (3)

3. Testing process

3.1 Instructions for input, detailing every step of the input process

3.2 Data to be recorded during the tests

4. Test cases (for each case)

4.1 Test case identification details

4.2 Input data and system settings

4.3 Expected intermediate results (if applicable)

4.4 Expected results (numerical, message, activation of equipment, etc.)

5. Actions to be taken in case of program failure/cessation

6. Procedures to be applied according to the test results summary

Test Implementation & Reporting

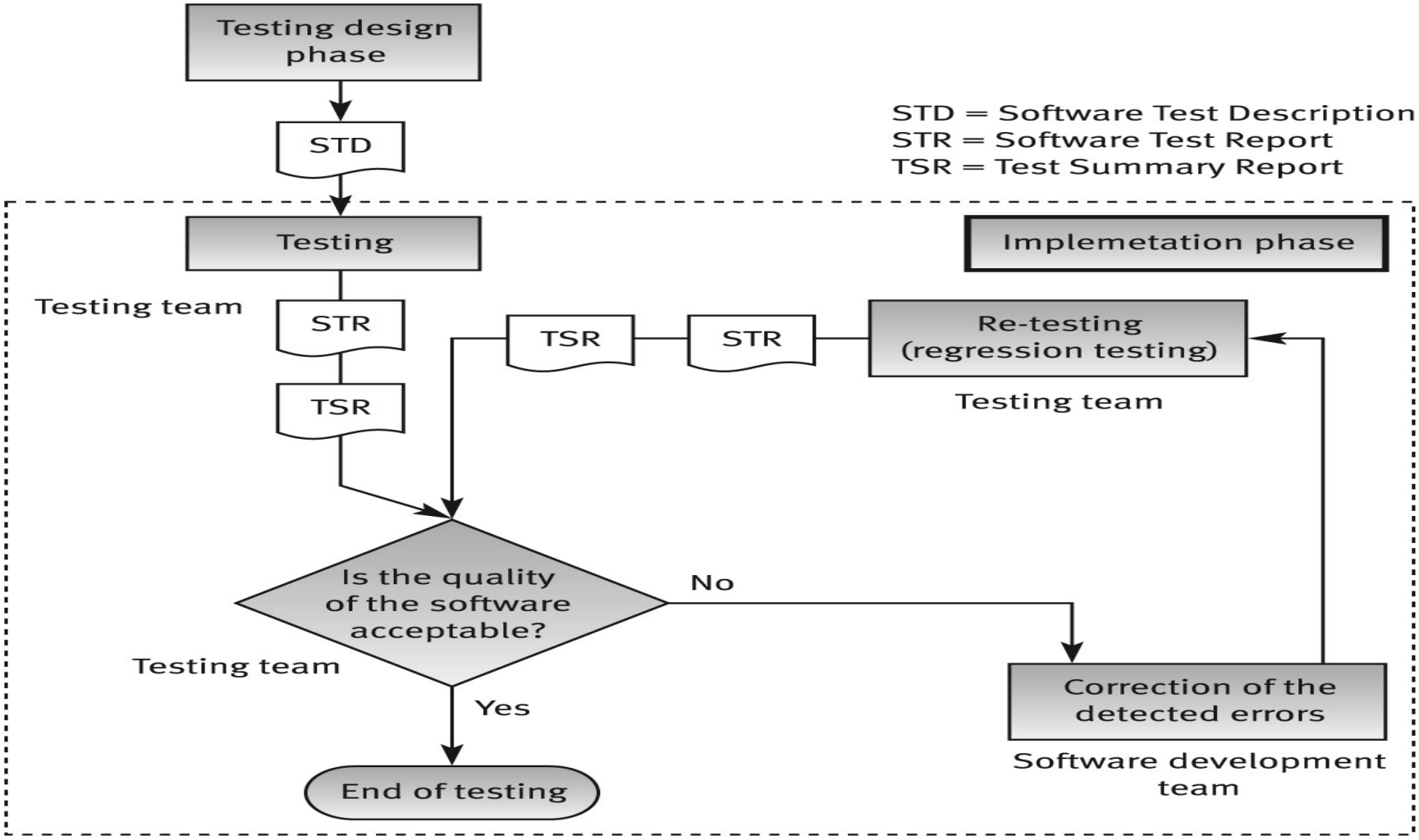
Test Implementation

- Really, this is just **running the tests**, **correction** of tests, **running regression** tests,
- Testing is done when the outcomes **satisfy the developers**.
- **When** are these **tests run**?? (time of day/ date??)

Regression Testing

- Need not test everything.
- Typically **re-test only those artifacts directly changed** and those providing inputs and outputs to these changed artifacts (modules).
- **Very often** **new** **errors** are introduced when changes are made.
- There's **always risk** in not testing everything… but these decisions must be made.
- **Results** of testing **are** **documented** in a test report.

Regression Testing Implementation Phase Activities



Software Test Report (STR) Template

1. Test identification, site, schedule and participation

1.1 The tested software identification (name, version and revision)

1.2 The documents providing the basis for the tests (name and version for each document)

1.3 Test site

1.4 Initiation and concluding times for each testing session

1.5 Test team members

1.6 Other participants

1.7 Hours invested in performing the tests

2. Test environment

2.1 Hardware and firmware configurations

2.2 Preparations and training prior to testing

Software Test Report (STR) Template (2)

3. Test results

3.1 Test identification

3.2 Test case results (for each test case individually)

4. Summary tables for total number of errors, their distribution and types

4.1 Summary of current tests

4.2 Comparison with previous results (for regression test summaries)

5. Special events and testers' proposals

5.1 Special events and unpredicted responses of the software during testing

5.2 Problems encountered during testing.

5.3 Proposals for changes in the test environment, including test preparations

5.4 Proposals for changes or corrections in test procedures and test case files