Computing and SE II

Chapter 12: Software Testing

Er-Yu Ding

Software Institute, NJU

Main Contents

1. What is software testing
2. Testing Level
3. Testing Strategies
4. Activities of software testing

1. What is software testing ---Software Correctness

- Software Correctness
  - A program P is considered *with respect to a specification* S, if and only if:
    - For each valid input, the output of P is in accordance with the specification S
- Software is never correct no matter which developing technique is used
- Any software must be validated and verified
  - Verification and Validation

1. What is software testing --- Verification and Validation

- Validation: is the system correct with respect to some specification?
- Verification: did we build the right system?
- V&V differences don’t matter
- V&V generally refers to any activity that attempts to ensure that the software will function as required

1. What is software testing --- If software not correct…

- Definitions
  - **Fault / Defect** – The cause of a failure, e.g., a missing or incorrect piece of code
  - **Error** – An intermediate unstable state
  - **Failure** – The manifested inability of the program to perform the function required, i.e., a system malfunction evidenced by incorrect output, abnormal termination or unmet time and space constraints.

- Relationship
  - A fault may remain undetected long time, until some event activates it.
  - If an when the error propagates to the output, it eventually causes the failure.
- Fault → Error → Failure
  - This chain can recursively iterate: a fault in turn can be caused by the failure of some other interacting system.

1. What is software testing ---Common V&V Activities

- Static Analysis Techniques –
  - Based solely on the (manual or automated) examination of project documentation of software models and code, and of other related information about requirements and design
  - Generally yields valid results, but may be weak in precision

- Dynamic Test Techniques –
  - Exercise the software in order to expose possible failures
  - Behavioral and performance properties of the program are also observed
  - Yields more precise results, but only holding for the examined executions

1. What is software testing ---Static Techniques

- Can be applied at the requirements phase; at the design phase; and during the implementation phase
- Traditional Static Techniques – heavily manual, error-prone, time consuming
  - **Software inspection** – The step-by-step analysis of the documents (deliverables) produced, against a compiled checklist of common and historical defects
  - **Software reviews** – The process by which different aspects of the work product are presented to project personnel (managers, users, customer etc) and other interested stakeholders for comment or approval
  - **Code reading** – The desktop analysis of the produced code for discovering typing errors that do not violate style or syntax
  - **Algorithm analysis and tracing** – The process in which the complexity of algorithms employed and the worst case, average-case and probabilistic analysis evaluations can be derived
- Static Analysis Techniques relying on use of Formal Methods

1. What is software testing ---Dynamic Techniques

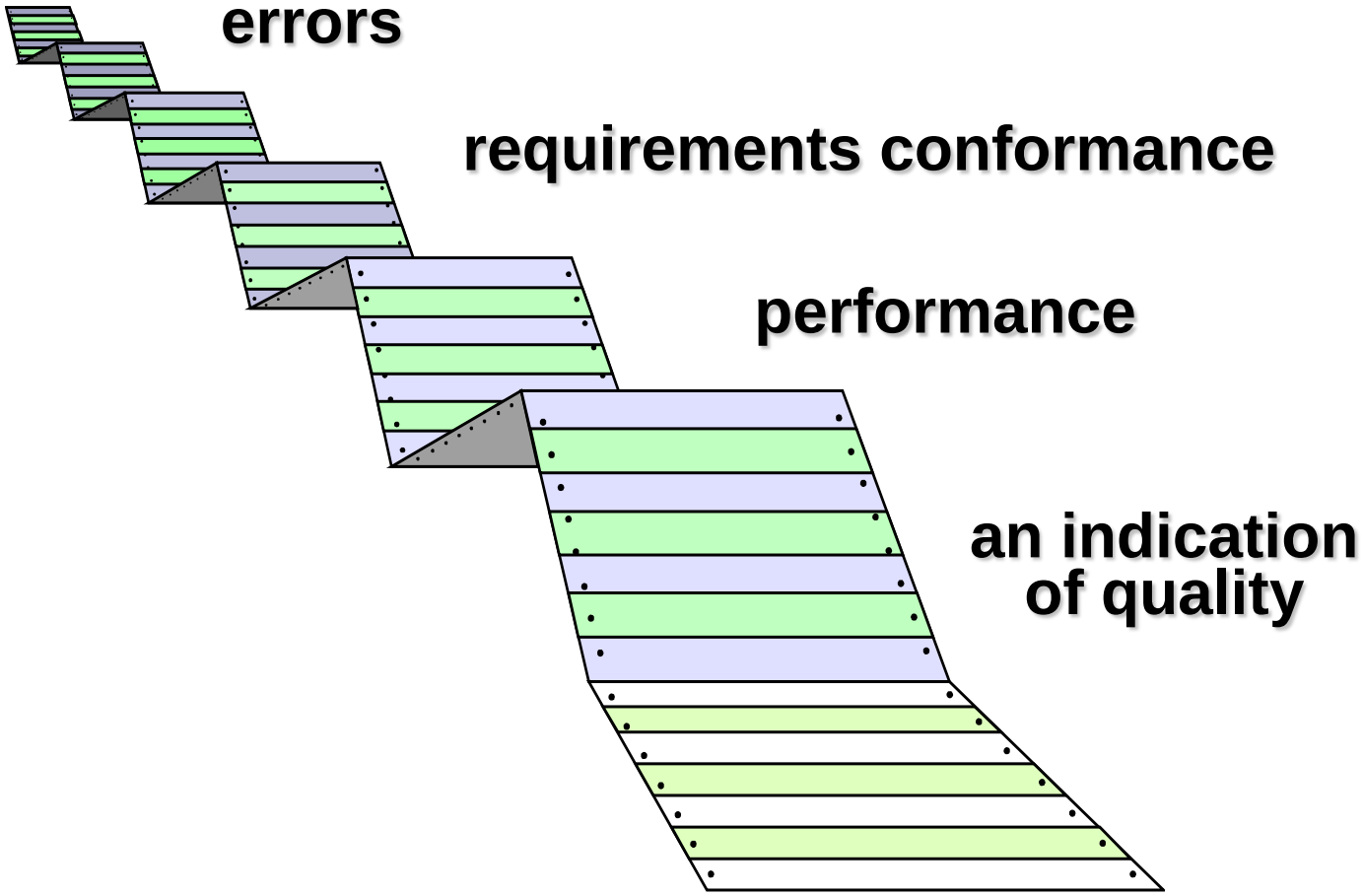
- Dynamic Techniques –
  - **Testing** – Based on the execution of the code on valued inputs (definition of the parameters and environmental conditions that characterize a system state must be included when necessary)
  - **Profiling** – A program profile records the number of times some entities of interest occur during a set of controlled executions

1. What is software testing

--- Software Testing

- Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.
  - – Def: The *dynamic* verification of the behavior of a program on a *finite* set of test cases, suitably *selected* from the usually infinite executions domain, against the *expected* behavior
  - Test Case: set of inputs and expected output
- Objective: to find errors
- Can detect the presence of defects, but not their absence

1. What is software testing --- What Testing Shows



1. What is software testing --- 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

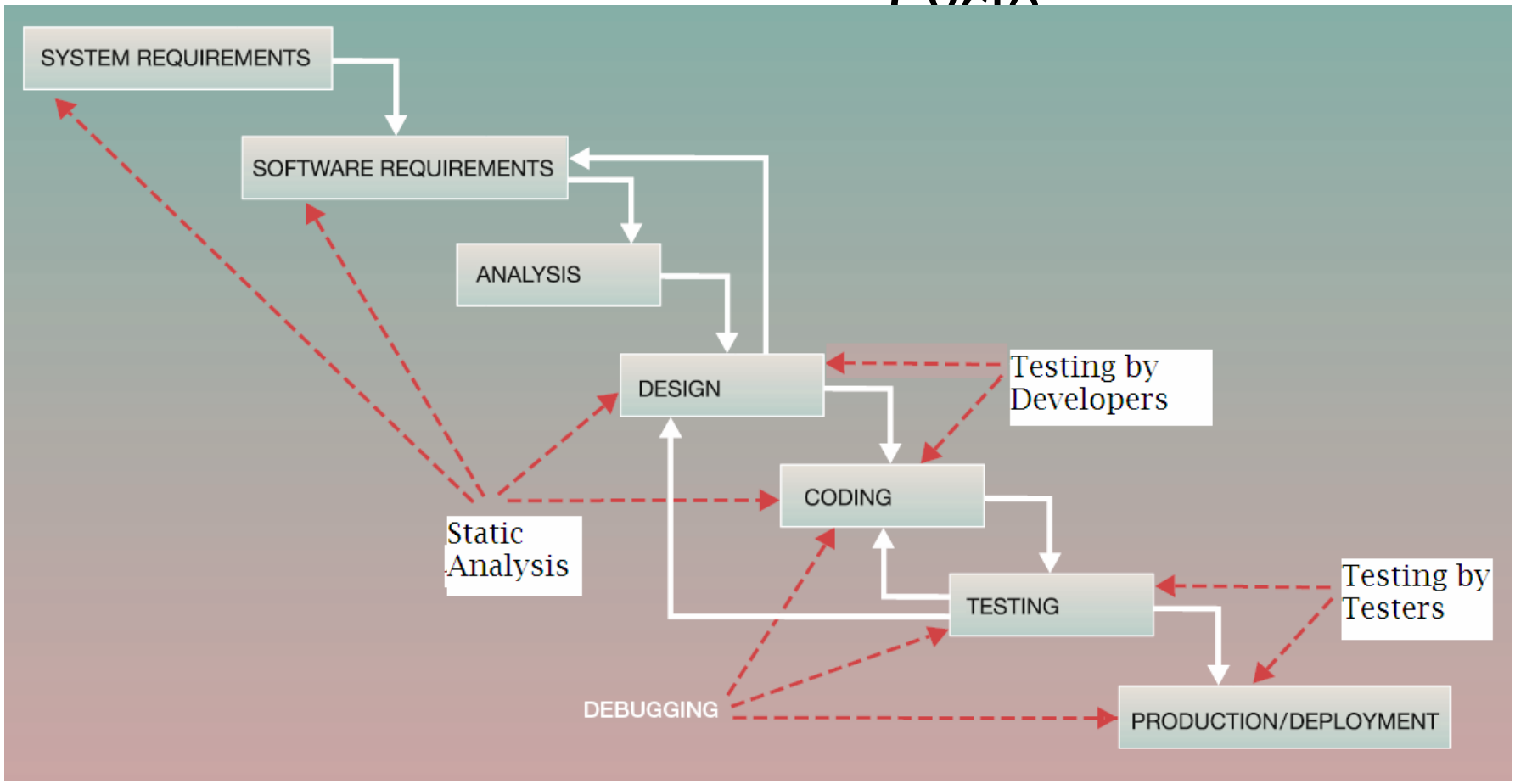
and, is driven by quality

1. What is software testing ---Debugging

- After errors detected
  - The process of debugging involves analyzing and possibly extending (with debugging statements) the given program that does not meet the specifications in order to find a new program that is close to the original and does satisfy the specifications

1. What is software testing

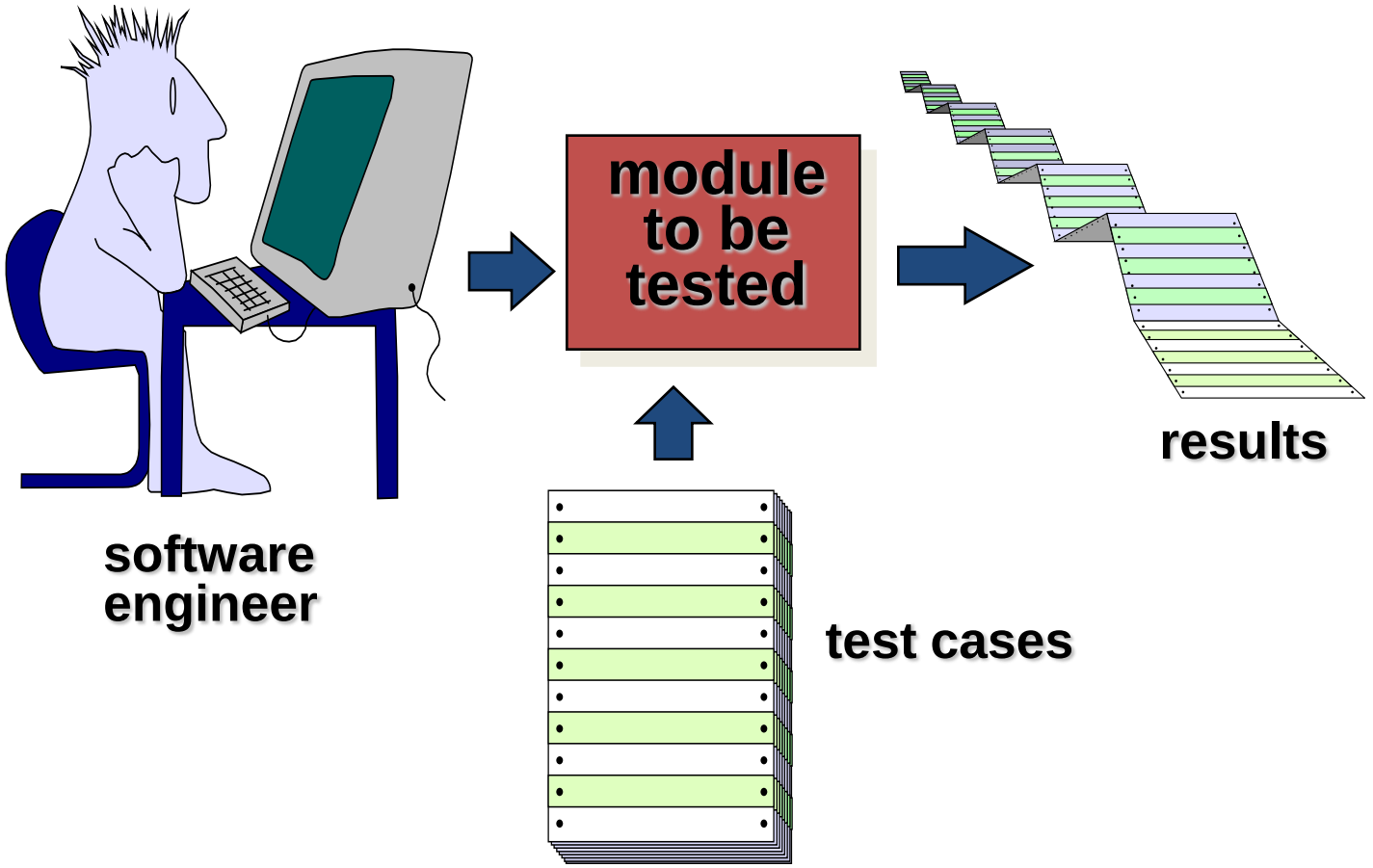
---V&V, Testing, Debugging and Life Cycle



2. Testing Level
--- Testing Level

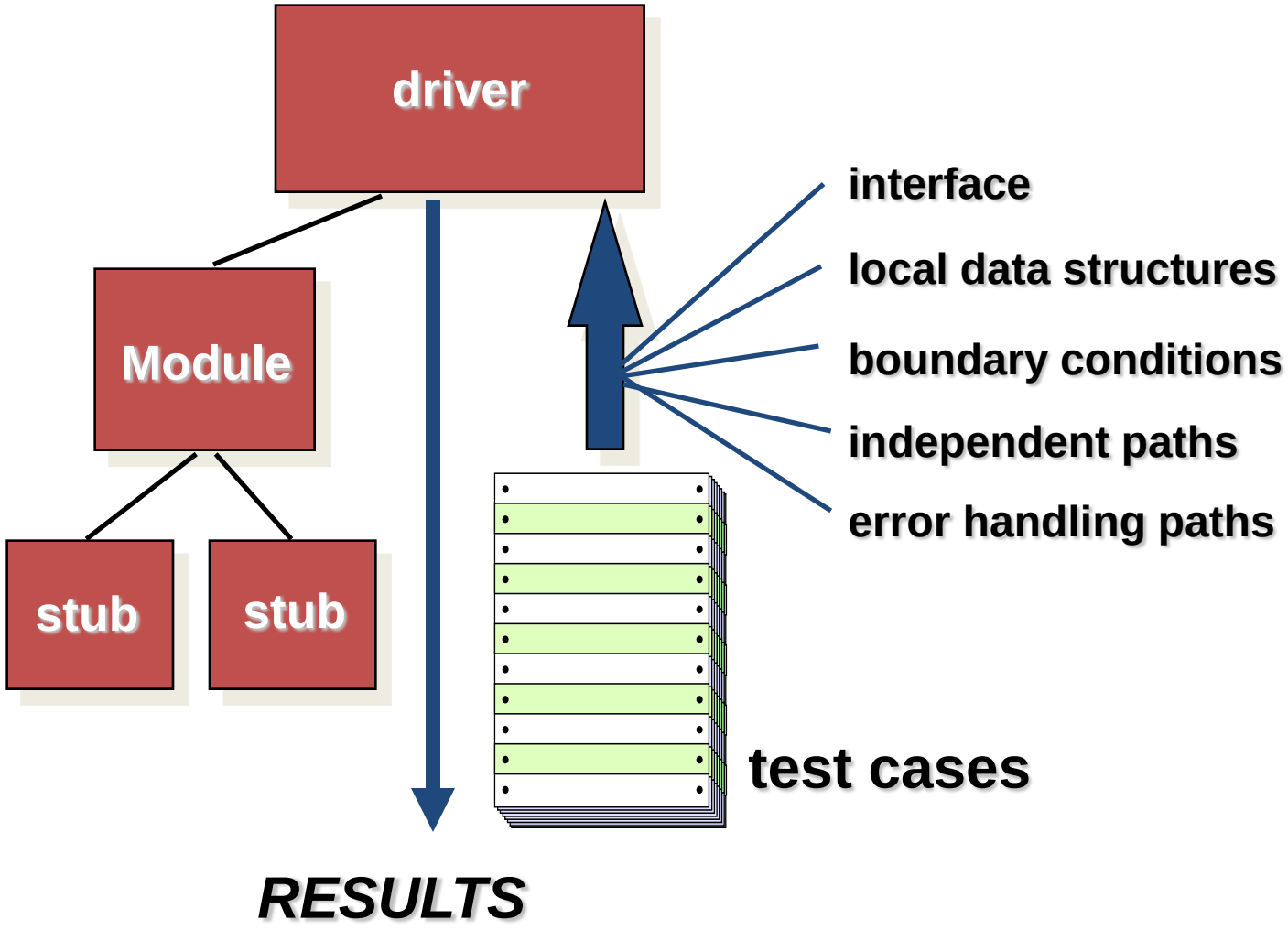
- Granularity levels
  - **Unit testing**: checking the behavior of single modules
  - **Integration testing**: checking the behavior of module cooperation.
  - **Acceptance testing**: the software behavior is compared with end user requirements
  - **System testing**: the software behavior is compared with the requirements specifications
  - **Regression testing**: to check the behavior of new releases

2. Testing Level
--- Unit Testing



2. Testing Level

--- Unit Test for Structural Software



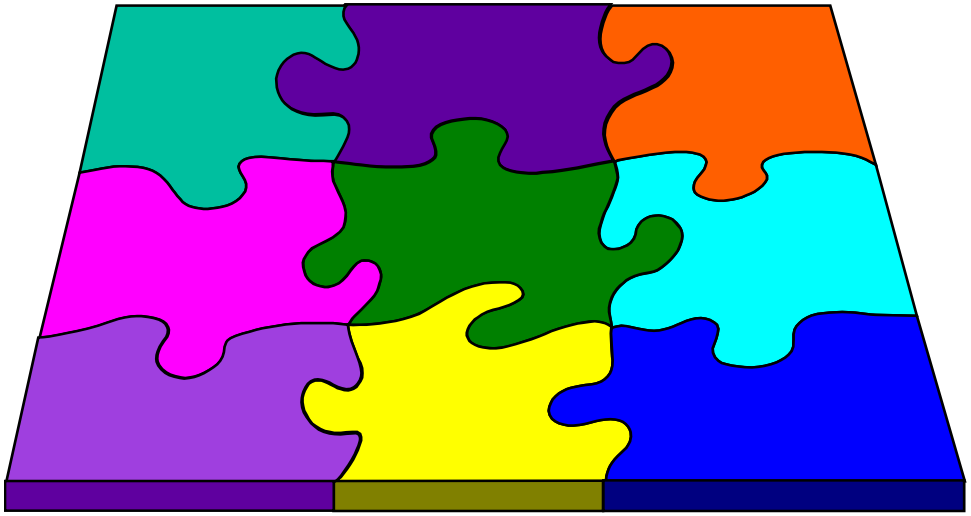
2. Testing Level
--- Unit Test for OO Software

- Testing strategy changes
  - the concept of the ‘unit’ broadens due to encapsulation
  - integration focuses on classes and their execution across a ‘thread’ or in the context of a usage scenario
    - operations within the class are tested
    - the state behavior of the class is examined

2. Testing Level
--- Integration Testing

Options:

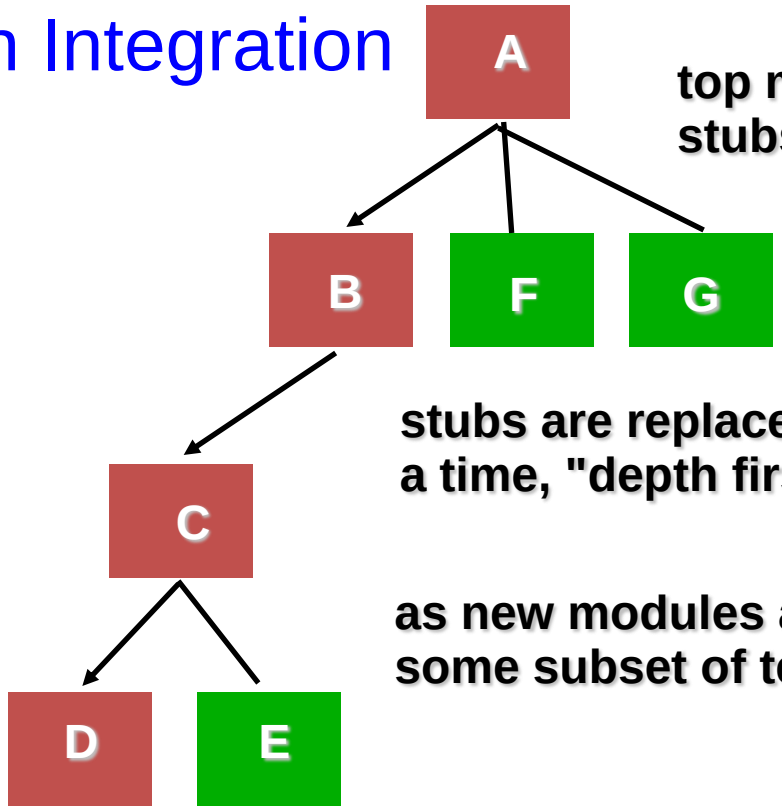
- **the “big bang” approach**
- **an incremental construction strategy**



2. Testing Level

--- Integration Testing for Structural Software

Top Down Integration



top module is tested with stubs

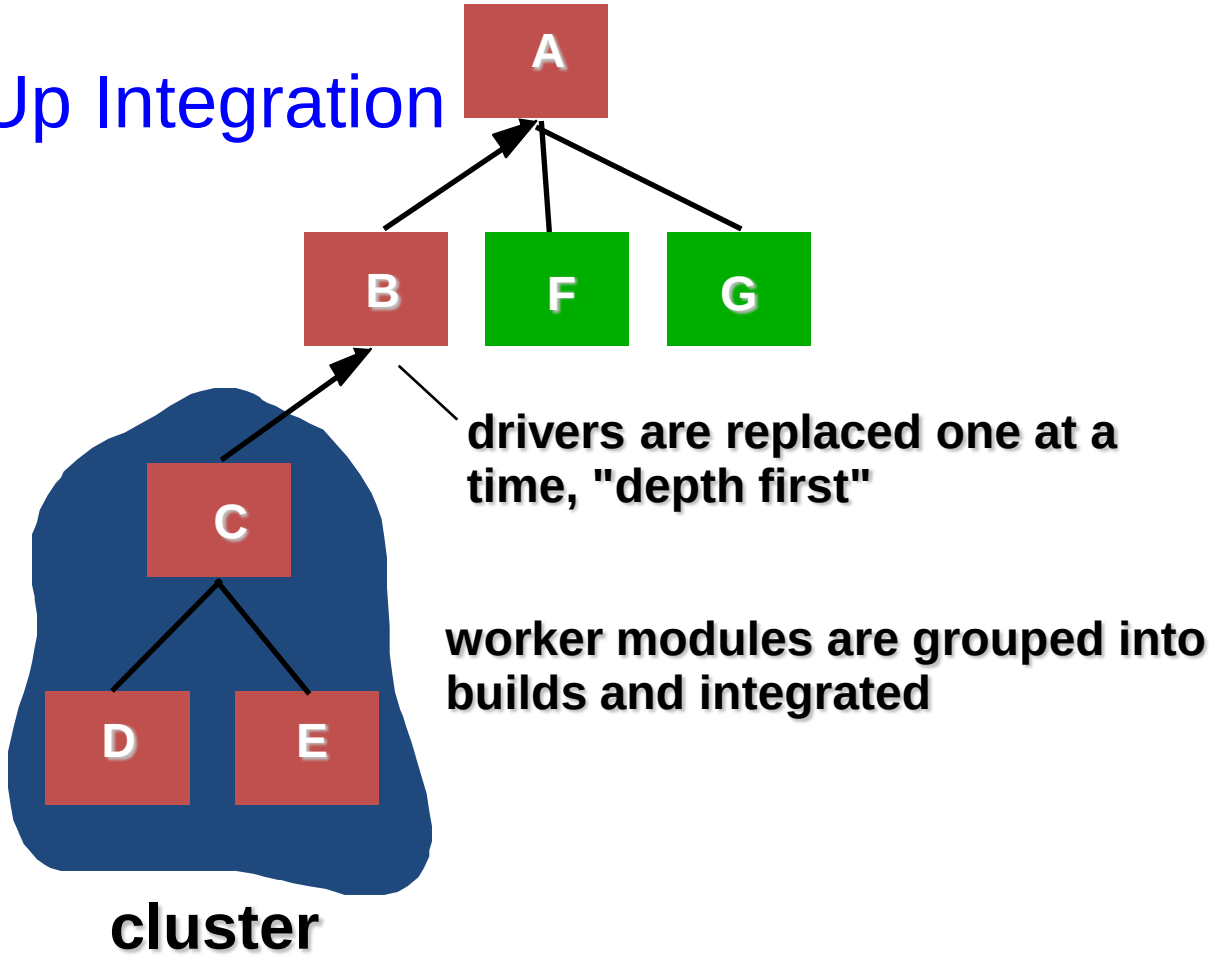
stubs are replaced one at a time, "depth first"

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

2. Testing Level

--- Integration Testing for Structural Software

Bottom-Up Integration



2. Testing Level

--- Integration Testing for OO Software

- integration applied three different strategies
  - thread-based testing—integrates the set of classes required to respond to one input or event
  - use-based testing—integrates the set of classes required to respond to one use case
  - cluster testing—integrates the set of classes required to demonstrate one collaboration

2. Testing Level
--- Integration Testing for All Software

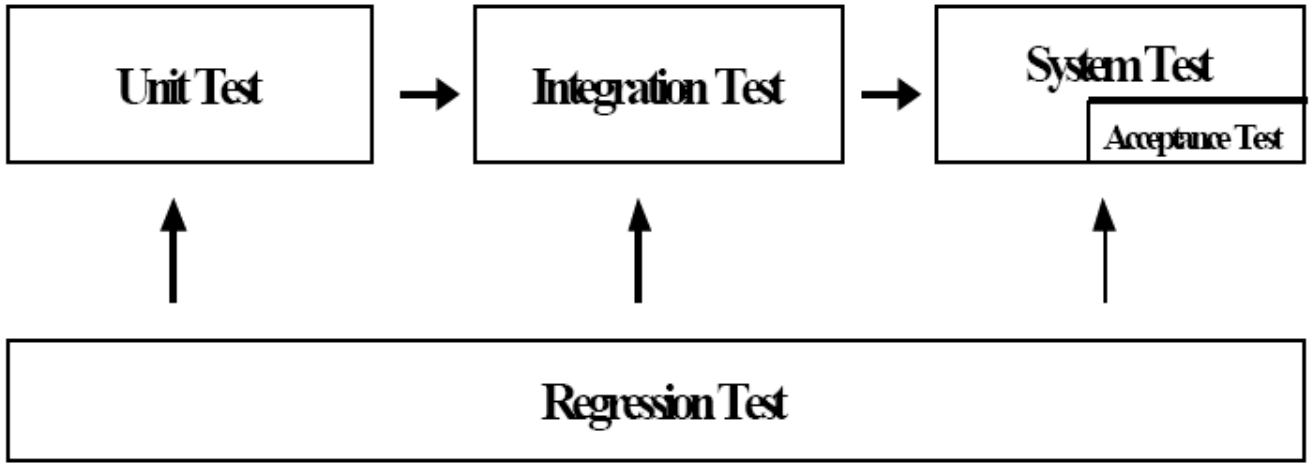
- Smoke Testing
  - A common approach for creating “daily builds” for product software
  - Smoke testing steps:
    - Software components that have been translated into code are integrated into a “build.”
      - A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
    - A series of tests is designed to expose errors that will keep the build from properly performing its function.
      - The intent should be to uncover “show stopper” errors that have the highest likelihood of throwing the software project behind schedule.
    - The build is integrated with other builds and the entire product (in its current form) is smoke tested daily.
      - The integration approach may be top down or bottom up.

2. Testing Level
---Acceptance Testing and System Testing

- Acceptance/Qualification Testing
  - Installation Testing
  - Alpha Testing
  - Beta Testing
- System testing
  - Recovery testing
  - Security testing
  - Stress testing
  - Performance Testing

2. Testing Level
---Regression Testing

- *regression test* is not a separate level of but may refer to the retesting of a unit, integration and system after modification, in order to ascertain that the change has



2. Testing Level
---An Example of Test Oracle

Detail Design	Unit Coding	Integration & Delivery	Maintenance
Design inspections	Code inspections	Integration test execution	Regression test execution
Generate oracles	Create scaffolding	System test execution	Revise regression test suite
Unit test planning	Unit test execution	Acceptance test execution	
Automated design analyses	Automated code analyses	Deliver regression test suits	
	Coverage analysis		

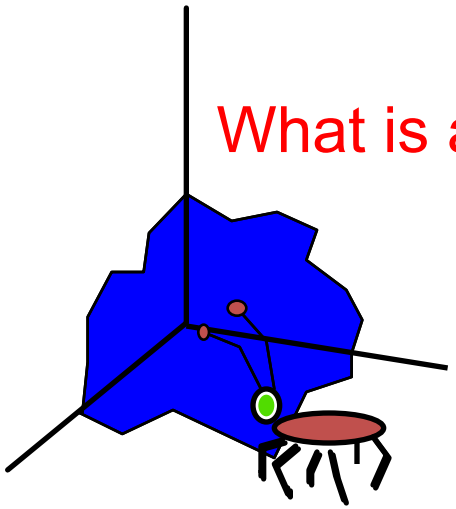
This method—a function that determines if the application has behaved correctly in response to a test action—is called a test oracle.

3. Testing strategies

--- Test Case Design

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

Boris Beizer



What is a “Good” Test?

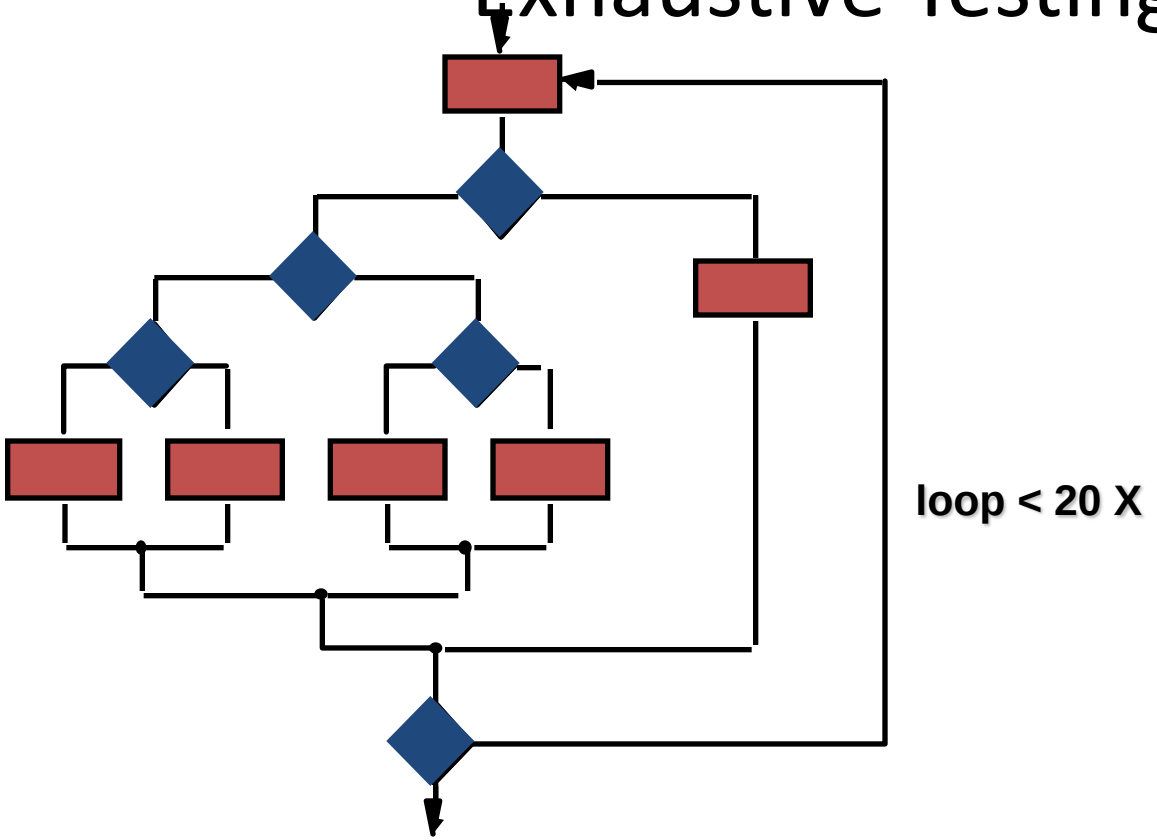
OBJECTIVE **to uncover errors**

CRITERIA **in a complete manner**

CONSTRAINT **with a minimum of effort and time**

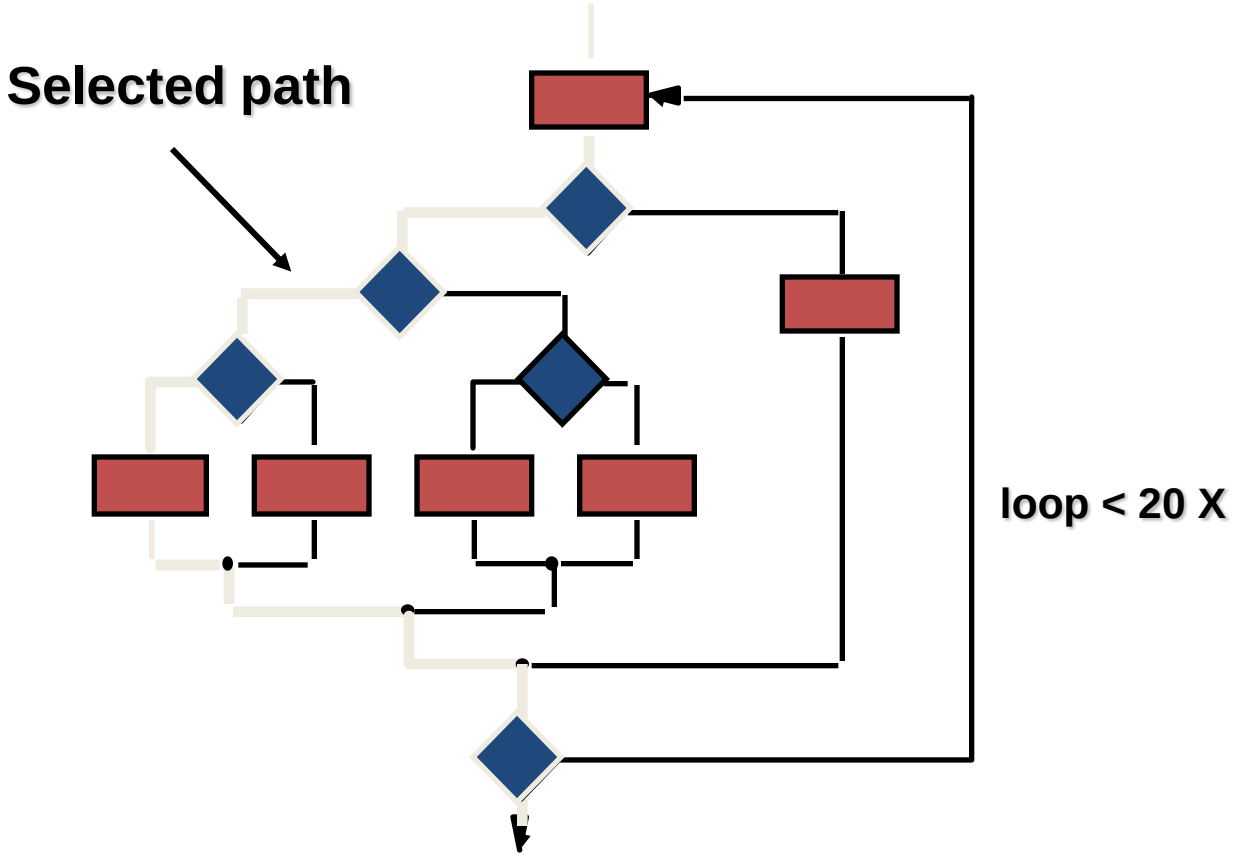
3. Testing strategies

--- Exhaustive Testing



There are 10^{14} possible paths! If we execute one test per millisecond, it would take 3,170 years to test this program!!

3. Testing strategies
--- Selective Testing



3. Testing strategies
---Testing Strategy

- Only exhaustive testing can show a program is free from defects. However, exhaustive testing is impossible,
- Develop strategy that maximize the chance of discovering bugs
  - Define the approach to be used in selecting tests
- **Systematic** exploration of a system characteristics
  - Testing should be an adaptive phase

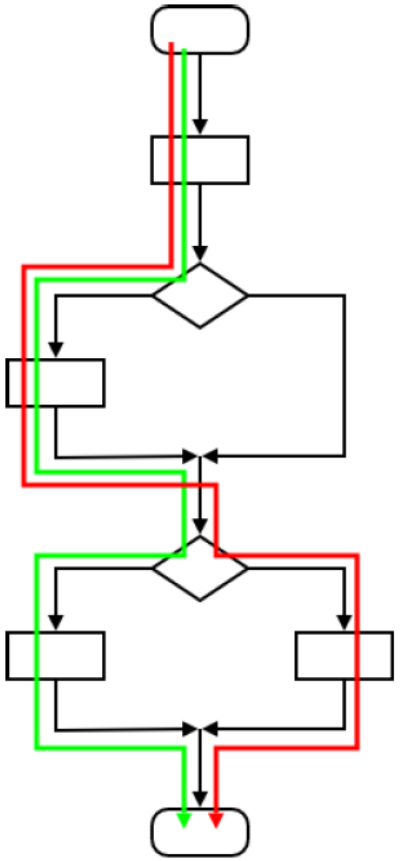
3. Testing strategies
--- Strategies for Test Case Selection

- Random Testing –
  - The test inputs are picked purely randomly from the whole input domain according to a specified distribution, i.e., after assigning to the inputs different “weights” (more properly probabilities)
- Structural Testing –Code-based testing or White-Box testing
  - Known as By monitoring code coverage one tries to exercise thoroughly all “program elements”
  - Code-based criteria should be more properly used as adequacy criteria
- Specification-Based Testing – Black-Box testing
  - Depending on how the program specifications are expressed, different techniques are possible - equivalence classes, boundary conditions, cause-effect graphs
- Others (Fault based, etc)
  - – based on classes of faults

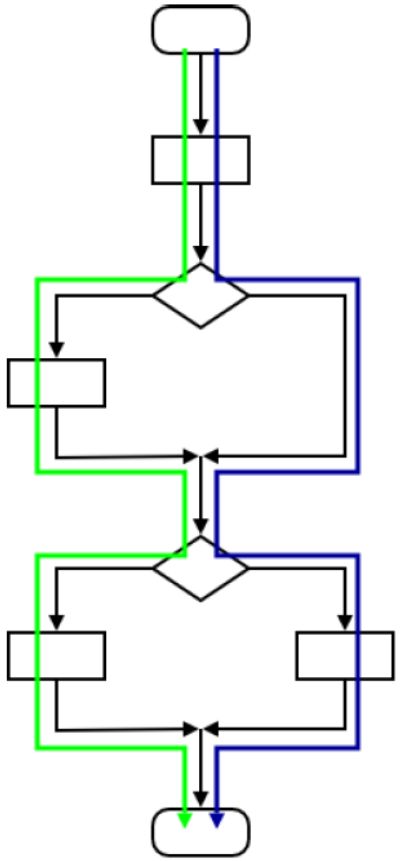
3. Testing strategies
---White-Box Testing

- All Statements Coverage:
  - All statements should be executed at least once
- All Paths Coverage
  - All paths should be executed at least once
- All Branches (Control Structure) Coverage
  - All branches should be executed at least once

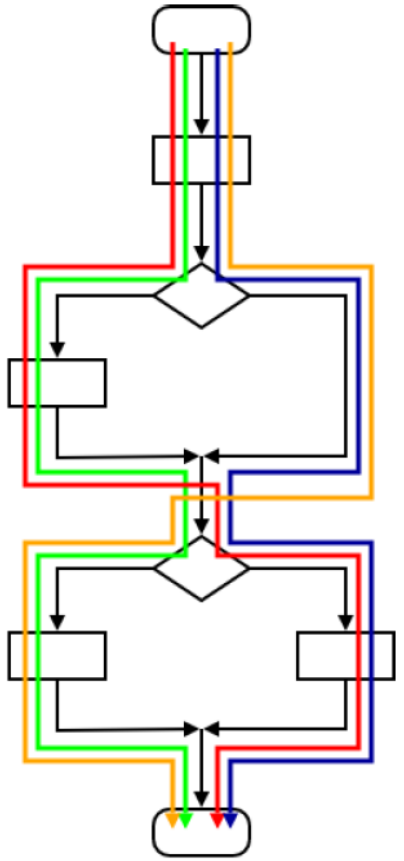
3. Testing strategies
---White-Box Testing



Statement Coverage
(aka line coverage)



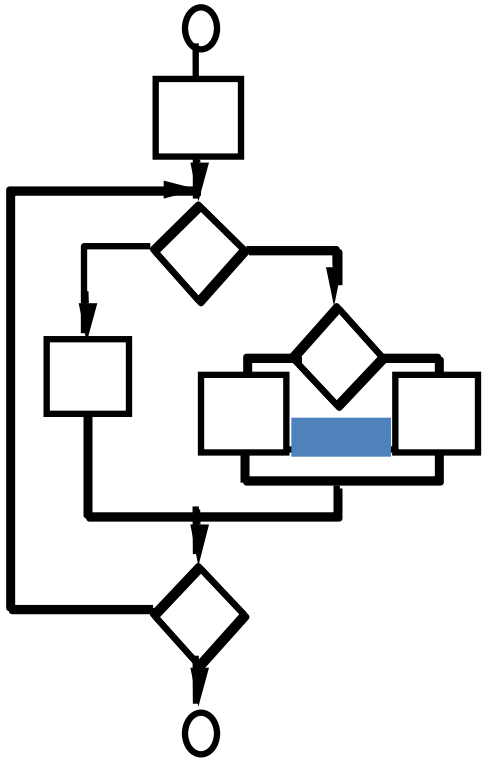
Branch Coverage
(aka condition coverage)



Path Coverage

3. Testing strategies

---White-Box Testing /All Paths Coverage



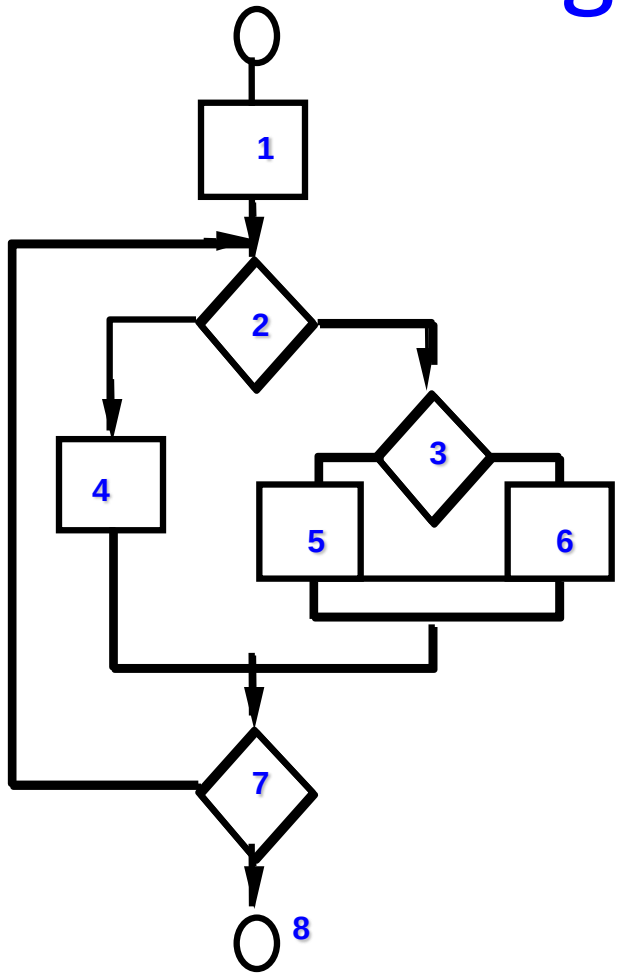
Basis Path Testing Notes

- **you don't need a flow chart, but the picture will help when you trace program paths**
- **count each simple logical test, compound tests count as 2 or more**
- **basis path testing should be applied to critical modules**

3. Testing strategies

--- White-Box Testing /All Paths Coverage

Basis Path Testing



Next, we derive the independent paths:

Since V(G) = 4, there are four paths

Path 1: 1,2,3,6,7,8
Path 2: 1,2,3,5,7,8
Path 3: 1,2,4,7,8
Path 4: 1,2,4,7,2,4,...7,8

Finally, we derive test cases to exercise these paths.

3. Testing strategies

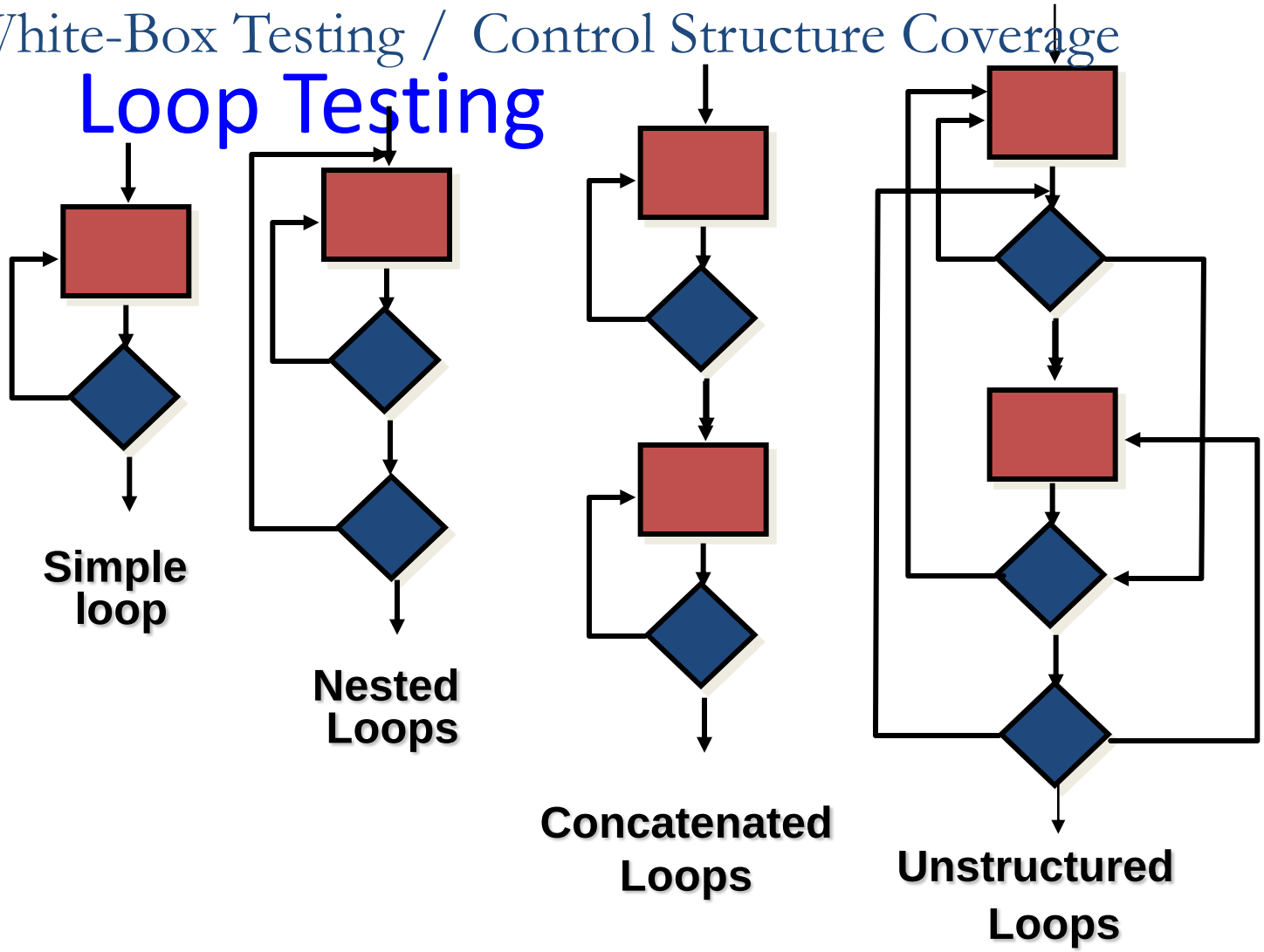
--- White-Box Testing / Control Structure Coverage

- Condition testing — a test case design method that exercises the logical conditions contained in a program module
- Data flow testing — selects test paths of a program according to the locations of definitions and uses of variables in the program

3. Testing strategies

--- White-Box Testing / Control Structure Coverage

Loop Testing

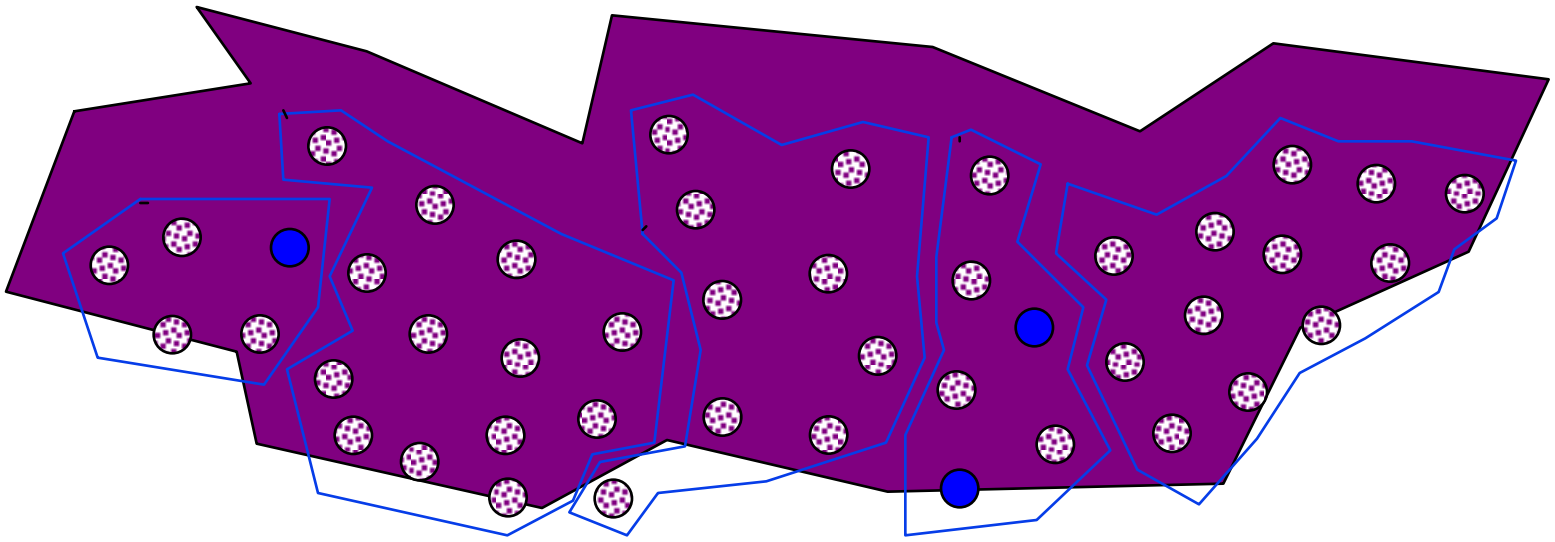


3. Testing strategies
--- Structural Coverage in Practice

- Statement and sometimes edge or condition coverage is used in practice
  - Simple lower bounds on adequate testing; may even be harmful if inappropriately used for test selection
- Additional control flow heuristics sometimes used
  - Loops (never, once, many), combinations of conditions

3. Testing strategies
--- Black-Box Testing / Partition Testing

- Basic idea: Divide program input space into (quasi-) equivalence classes



3. Testing strategies
--- Black-Box Testing / Boundary Value Analysis

- Premise: Bugs tend to lurk around the edges

- Those combinations of values that are “close” (actually on, above and beneath) the borders of the equivalence classes identified both in the input and the output domains.

3. Testing strategies

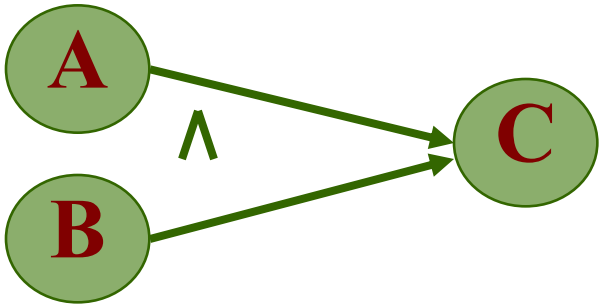
--- Black-Box Testing / Cause-effect graphs

- Captures the relationships between specific combinations of inputs (causes) and outputs (effects)
  - Deals with specific cases,
  - Avoids combinatorial explosion
  - Explore combinations of possible inputs
- Causes/effects are represented as nodes of a cause effect graph
- The graph also includes a number of intermediate nodes linking causes and effects

3. Testing strategies

--- Black-Box Testing / Cause-Effect Graphs

If A then B (identity)



If (A and B)then C

3. Testing strategies
--- White vs black box

Black box

- it depends on the specification notation
- it scales up (different techniques at different granularity levels)
- it cannot reveal code bases testing (same specification implemented with different modules)

White box

- it is based on control or data flow coverage
- it does not scale up (mostly applicable at unit and integration testing level)
- it cannot reveal missing path errors (part of the specification that is not implemented)

4. Activities of software testing

- Testing Design
- Testing Execution
- Testing Result Documentation
- Testing Results Evaluation
- Testing Management

4. Activities of software testing ---Testing Design

- Developing a test strategy which efficiently meets the needs
  - identifying testing levels and methods, techniques and tools to be used
- A test plan states:
  - What the items to be tested are, at what level they will be tested, what sequence they are to be tested in, how the test strategy will be applied to the testing of each item, and describes the test environment
  - Acceptance & system test planning (requirements engineering)
  - Integration & unit test planning (architectural design)
- Test case generation
  - Specifying a set of test cases or test paths for each item to be tested at that level
  - Implementing a set of test cases as a test procedure

4. Activities of software testing --- Testing Execution

- Selection testing tools
  - The usage of appropriate tools can therefore alleviate the burden of clerical, tedious operations, and make them less error-prone, while increasing testing efficiency and effectiveness.
- Launching the tests
  - Forcing the execution of the test cases (manually or automatically)

4. Activities of software testing --- Testing Result Documentation

- The outputs of each test execution should be recorded in a test results file.
- Documents in testing
  - Testing Design Document
    - Test Plan
    - Test Design Specification
    - Test Case Specification
    - Test Procedure Specification
  - Testing Result document
    - Test Log
    - Test Incident or Problem Report.

4. Activities of software testing ---Testing Results Evaluation

- Evaluation of the Program under Test
- Evaluation of the Test performed
- Measures for monitoring the testing process

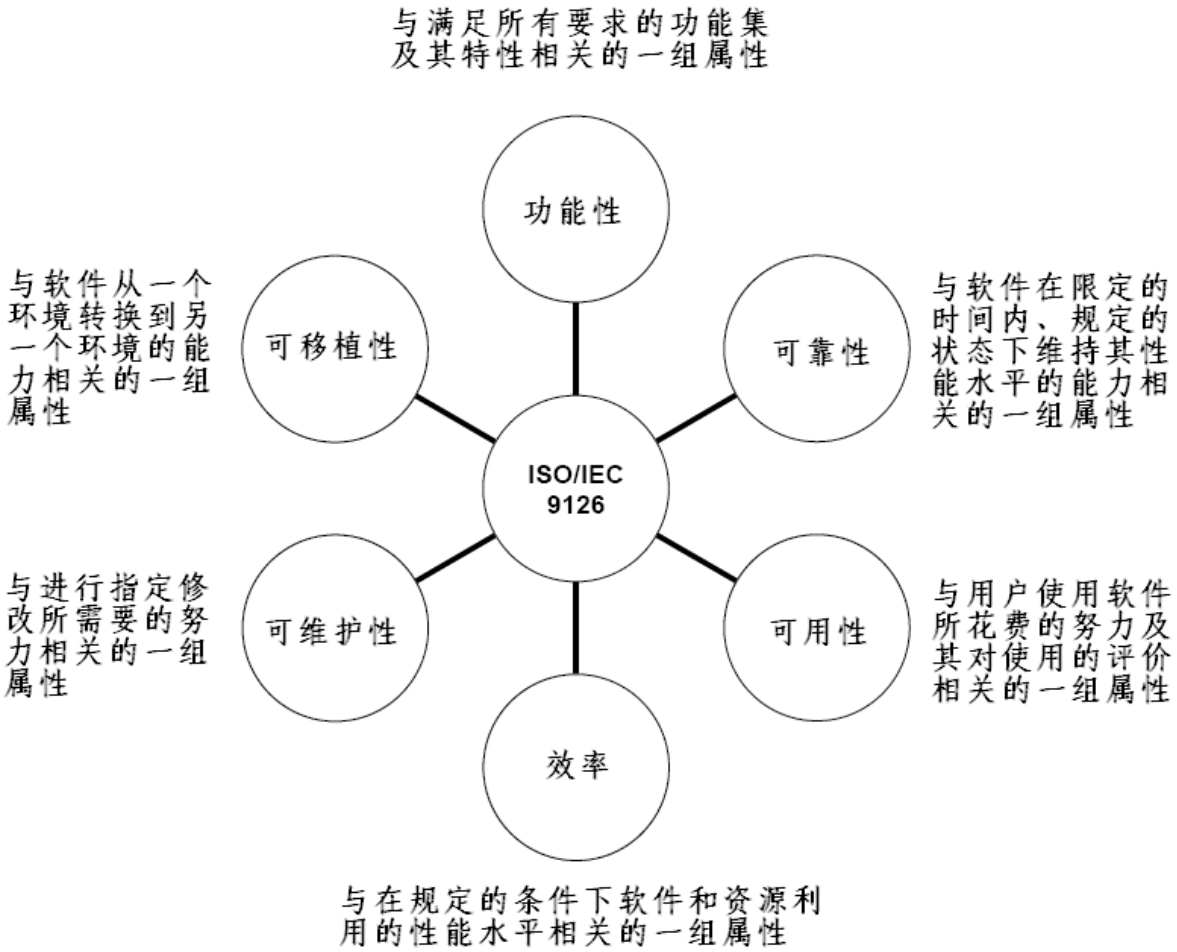
4. Activities of software testing --- Testing Management

- Different management activities
  - Scheduling the timely completion of tasks
  - Estimation of the effort and the resources needed to execute the tasks
  - Quantification of the risk associated with the tasks
  - Effort/Cost estimation
  - Quality control measures to be employed

4. Activities of software testing --- How to decide when to stop testing

- The main problem for managers
- When resources (time and budget) are over (Wrong)
  - no information about the efficacy of the test
  - BUT… resource constraints must be taken into account
- When some coverage is reached (Wrong)
  - no assurance of software quality
  - it can be a reasonable and objective criterion
  - It can be (partially) automated
- When quality criterion is reached (Right)
  - Quality criterion is established according to

4. Activities of software testing ——What is software quality ?



4. Activities of software testing

---An Example of Test Oracle

Detail Design	Unit Coding	Integration & Delivery	Maintenance
Design inspections	Code inspections	Integration test execution	Regression test execution
Generate oracles	Create scaffolding	System test execution	Revise regression test suite
Unit test planning	Unit test execution	Acceptance test execution	
Automated design analyses	Automated code analyses	Deliver regression test suite	
	Coverage analysis		

The End

- Recommend paper
  - 《A Brief Essay on Software Testing》
- The next lecture
  - Software maintenance