

ICS 121 Topic 11:
Testing

Definitions
Fundamental Questions
Test Criteria
Regression Testing during Maintenance
V&V with Formal Specifications

Verification and Validation

```

    graph TD
        IR[Informal Requirements] --> FS[Formal Specification]
        FS --> SI[Software Implementation]
        SI -- Verification --> FS
        SI -- Validation --> IR
    
```

Verification: is implementation consistent with requirements specification?
Validation: does the system meet the customer's/user's needs?

Software Quality: assessment by V&V

- Software process must include verification and validation to measure product qualities
 - correctness, reliability, robustness
 - efficiency, usability, understandability
 - verifiability, maintainability
 - reusability, portability, interoperability, extensibility
 - real-time, safety, security, fault tolerance, accuracy
- Products can be improved by improving the process by which they are developed and assessed

Failures, Errors, Faults

- Failure: incorrect/unexpected behavior or output
 - incident is the symptoms revealed by execution
 - failures are usually classified
- Potential Failure: incorrect internal state
 - sometimes also called an error, state error, or internal error
- Fault: anomaly in source code
 - may or may not produce an error
 - a “bug”
- Error: inappropriate development action
 - action that introduced a fault

Examples: Faults, Errors, and Failures

```

    graph TD
        N1(1: input A,B) --> N2(2: A:=0?)
        N2 --> N3(3: C:=0)
        N2 --> N4(4: C:=A*B)
        N3 --> N5(5: B>0?)
        N4 --> N5
        N5 --> N6(6: C:=C*(A+2))
        N5 --> N7(7: X:=A+B)
        N6 --> N8(8: output X)
        N7 --> N8
    
```

- Suppose node 6 should be $X := C*(A+2*B)$
 - Failure/Error-less fault:
 - executing path (1,2,4,5,7,8) will not reveal this fault because 6 is not executed
 - nor will executing path (1,2,3,5,6,8) because $C = 0$
- Need to make sure proper test cases are selected
 - the definitions of C at nodes 3 and 4 both affect the use of C at node 6
 - executing path (1,2,4,5,6,8) will reveal the failure, but only if $B \neq 0$

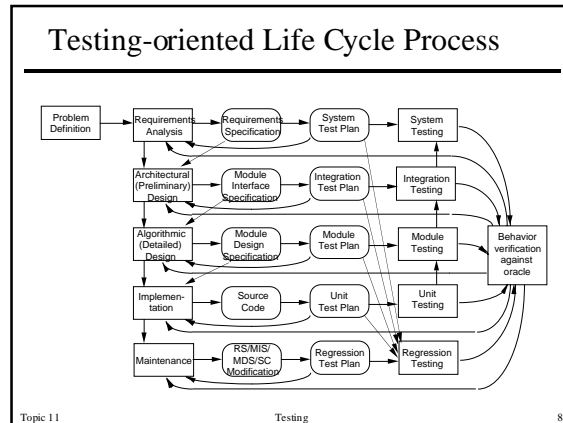
Software Testing

- Exercising a system [component]
 - on some predetermined input data
 - capturing the behavior and output data
 - comparing with test oracle
 - for the purposes of
 - identifying inconsistencies
 - verifying consistency between actual results and specification
 - to provide confidence in consistency with requirements and measurable qualities
 - to demonstrate subjective qualities
 - validating against user needs
- Limitations
 - only as good as the test data selected
 - subject to capabilities of test oracle

Levels of Testing

- Unit testing: testing of code unit (subprogram, module, subsystem)
 - Usually requires use of test drivers
- Integration testing: testing of interfaces between integrated units
 - Incremental or “big bang”
- System testing: testing complete system for satisfaction of requirements
- Regression testing: testing after change

Topic 11 Testing 7



Fundamental Testing Questions

- Test Criteria: What should we test?
- Test Oracle: Is the test correct?
- Test Adequacy: How much is enough?
- Test Process: Is our testing effective?

How to make the most of limited resources?

Topic 11 Testing 9

Practical Issues

- Purpose of testing
 - Fault detection
 - High assurance of reliability
 - Performance/stress/load
 - Regression testing of new versions
- Conflicting considerations
 - safety, liability, risk, customer satisfaction, resources, schedule, market windows and share
- Test Selection is a sampling technique
 - choose a finite set from an infinite domain

Topic 11 Testing 10

Test Case

- Specification of
 - identifier
 - test items
 - input and output specs
 - environmental requirements
 - procedural requirements
- Augmented with history of
 - actual results
 - evaluation status
 - contribution to coverage

Topic 11 Testing 11

Test Criteria

- Testing must select a subset of test cases that are likely to reveal failures
- Test Criteria provide the guidelines, rules, strategy by which test cases are selected
 - actual test data
 - conditions on test data
 - requirements on test data
- Equivalence partitioning is the typical approach
 - a test of any value in a given class is equivalent to a test of any other value in that class
 - if a test case in a class reveals a failure, then any other test case in that class should reveal the failure
 - some approaches limit conclusions to some chosen class of faults and/or failures

Topic 11 Testing 12

Test Oracle

- A test oracle is a mechanism for verifying the behavior of test execution
 - extremely costly and error prone to verify
 - oracle design is a critical part of test planning
- Sources of oracles
 - input/outcome oracle
 - tester decision
 - regression test suites
 - standardized test suites and oracles
 - gold or existing program
 - formal specification

Topic 11 Testing 13

Test Adequacy

- Coverage metrics
 - when sufficient percentage of the program structure has been exercised
- Empirical assurance
 - when failures/test curve flatten out
- Error seeding
 - percentage of seeded faults found is proportional to the percentage of real faults found
- Independent testing
 - faults found in common are representative of total population of faults

Topic 11 Testing 14

Functional and Structural Testing

- Functional Testing
 - Test cases selected based on specification
 - Views program/component as black box
- Structural Testing
 - Test cases selected based on structure of code
 - Views program /component as white box (also called glass box testing)

*Can do black-box testing of **program** by doing white-box testing of **specification***

Topic 11 Testing 15

Black Box vs. White Box Testing

"BLACK BOX" TESTING

"WHITE BOX" TESTING

Topic 11 Testing 16

Structural (White-Box) Test Criteria

- Criteria based on
 - control flow
 - data flow
 - expected faults
- Defined formally in terms of flow graphs
- Metric: percentage of coverage achieved
- Adequacy based on metric requirements for criteria

*Objective: **Cover the software structure***

Topic 11 Testing 17

Flow Graphs

- Control Flow
 - The partial order of statement execution, as defined by the semantics of the language
- Data Flow
 - The flow of values from definitions of a variable to its uses

Graph representation of control flow and data flow relationships

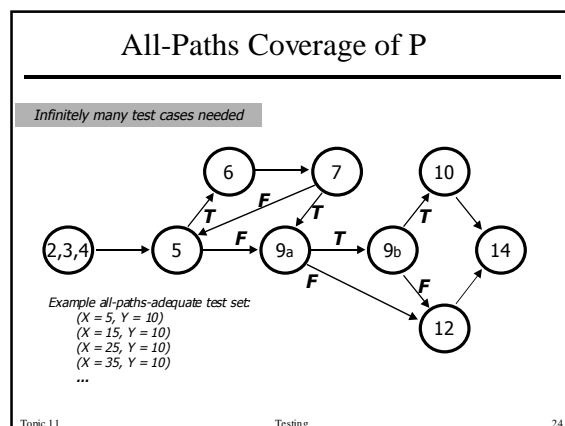
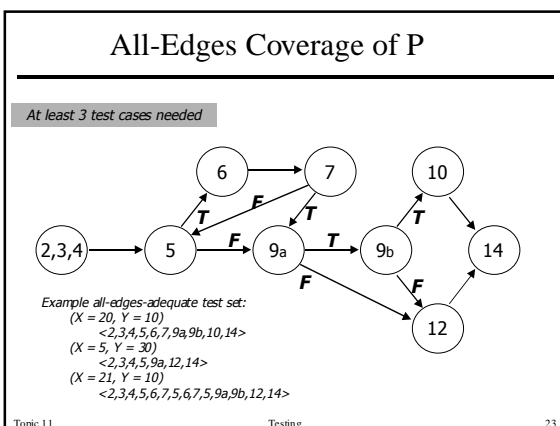
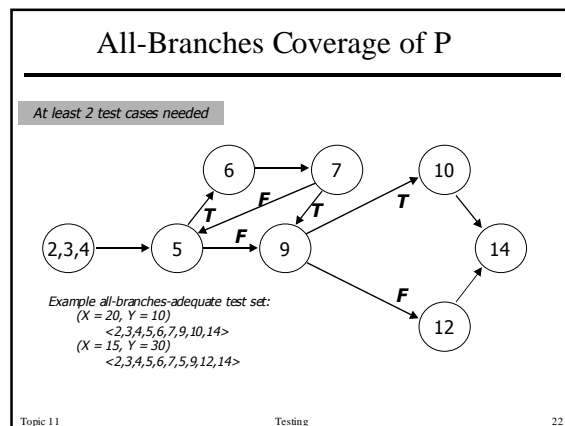
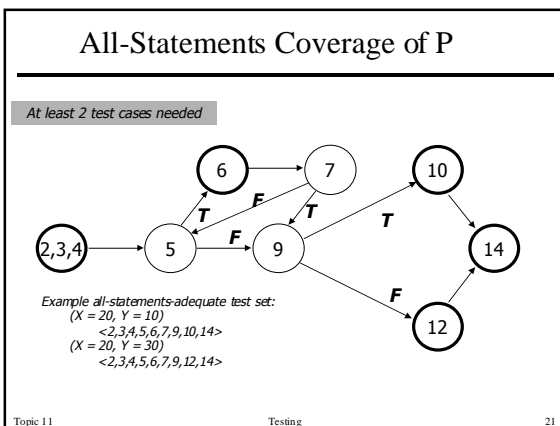
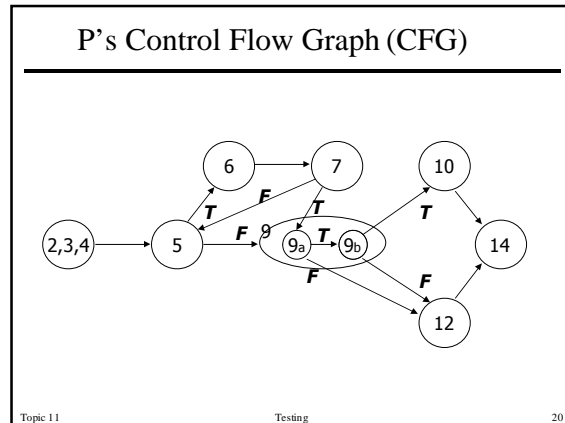
Topic 11 Testing 18

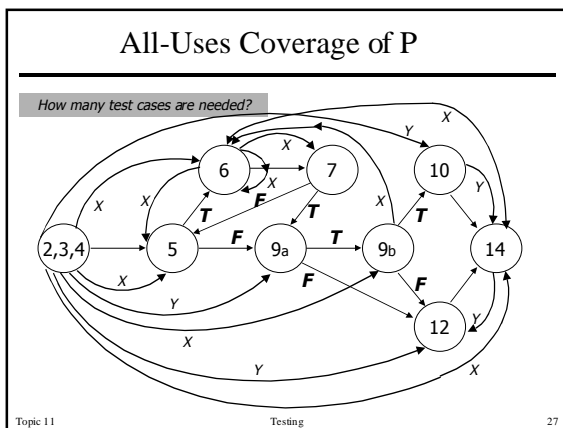
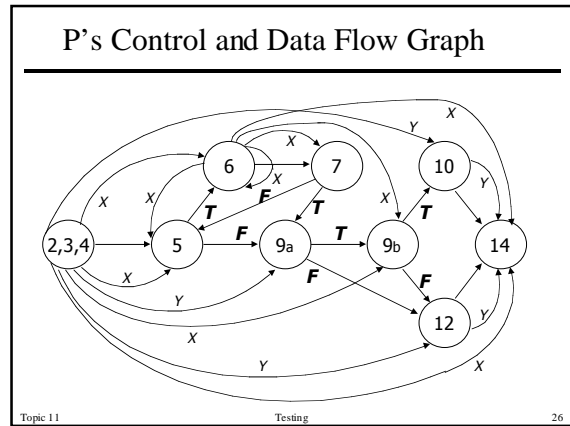
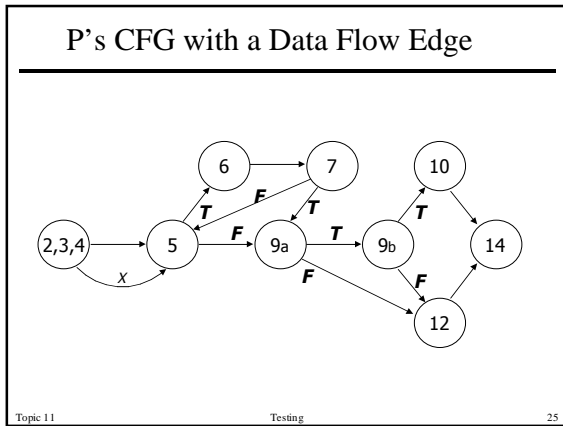
A Sample Program to Test

```

1  function P return INTEGER is
2  begin
3    X, Y: INTEGER;
4    READ(X); READ(Y);
5    while (X > 10) loop
6      X := X - 10;
7      exit when X = 10;
8    end loop;
9    if (Y < 20 and then X mod 2 = 0) then
10     Y := Y + 20;
11   else
12     Y := Y - 20;
13   end if;
14   return 2*X + Y;
15 end P;
    
```

Topic 11 Testing 19

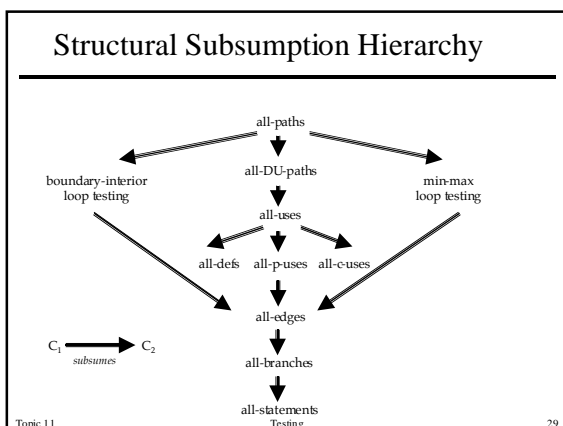




Subsumption of Criteria

- C1 subsumes C2 if any C1-adequate T is also C2-adequate
 - But some T1 satisfying C1 may detect fewer faults than some T2 satisfying C2

Topic 11 Testing 28



Software Maintenance

- Typically > 60% of development costs
- Most modifications are enhancements and new features
- Testing costs dominate
 - Many systems require labor-intensive, manual testing
- Need to find ways of reducing cost

Topic 11 Testing 30

Regression Testing

- Permanent suite of test cases
 - Saves effort creating test cases
 - Provides record of existing functionality
- Add new test cases and delete obsolete ones when necessary

Ensure that changes made during maintenance do not destroy existing functionality

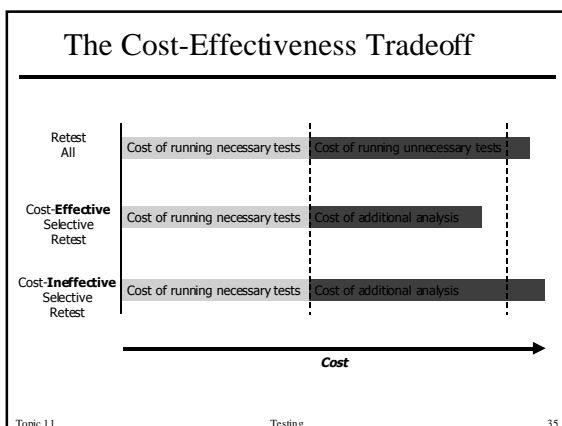
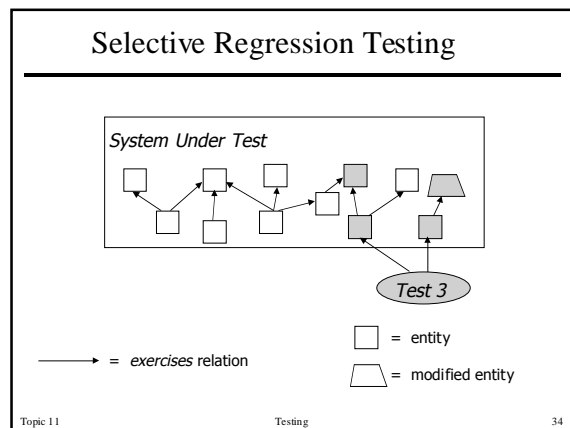
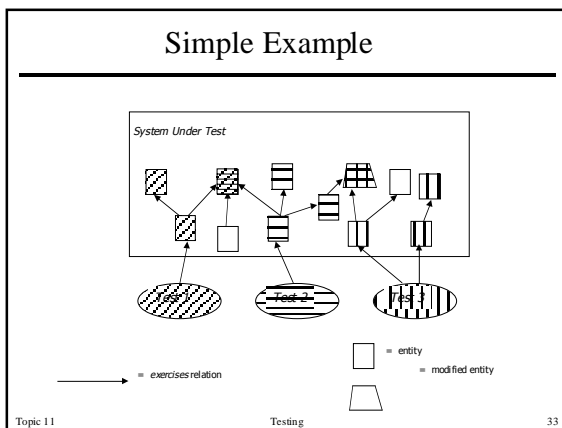
Topic 11 Testing 31

Selective Regression Testing

- Analyze relationship between the test cases and the software entities they cover
- Use information about changes to select test cases for new version

Cost reduction:
Select minimal subset of regression test suite that tests the changes

Topic 11 Testing 32



The Reality of Software Testing

- One of the most widely-used, practical class of techniques for revealing faults in software
- An area of software engineering showing the greatest gap between theory and practice

Topic 11 Testing 36

Formal Verification

- Techniques for proving consistency between two software descriptions
 - subject to assumptions of proof system
 - only as good as formal specification
 - to prove consistency of specification
 - to prove correctness of implementation
 - does not show other qualities, especially the subjective ones
 - has not been shown to scale up to large-scale software systems
 - only informal techniques for validating against user needs

Topic 11 Testing 37

Verification with Formal Specifications

Topic 11 Testing 38

Testing with Formal Specifications

Topic 11 Testing 39

Formal Specification-Based Testing

- Formalize functional testing for formal specifications
- Extend implementation-based techniques to be applicable to formal specifications
- Dependent on the specification language
 - Model-based specifications: test cases selected to structurally cover specification
 - Algebraic specifications: test cases selected to test against axioms or axioms tested by interpretation on test data
 - State-based specifications: test cases selected to cover operations' state changes

Specification-based testing should augment implementation-based testing, not replace it

Topic 11 Testing 40

Something to think about: Testing Process/Integration

- It is generally accepted that no single technique is sufficient to verify and validate software
- How do you choose?
 - strengths and weaknesses
 - integration issues
 - incremental issues

Topic 11 Testing 41