Module – II
Software Engineering Development Practices

Software Testing

Content Area 1

Software Testing Fundamentals

Testing-related terminology - I

- **SWEBOK 2004**: Software testing is "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."
  - **dynamic**: testing done by executing the program on (valued) inputs
  - **finite**: the whole test set can generally be considered infinite. Testing is a trade-off between the quality of the testing and the resources available
  - **selected**: "test set" selection based on risk analysis and test engineering
  - **expected**: must be possible to decide whether the observed outcomes of program execution are acceptable.

Testing-related terminology - II

- **IEEE Std. 610.12-1990**: defines testing as "the process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component."
- Other testing related terms describing a malfunction:
  - **Fault**: An incorrect step, process, or data definition in a computer program.
  - **Error**: The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
  - **Failure**: The inability of a system or component to perform its required functions within specified performance requirements.
  - **Mistake**: A human action that produces an incorrect result.

Discussion question

- **An incorrect step, process, or data definition in a computer program is a**

  a. Failure

  b. Mistake

  c. Fault

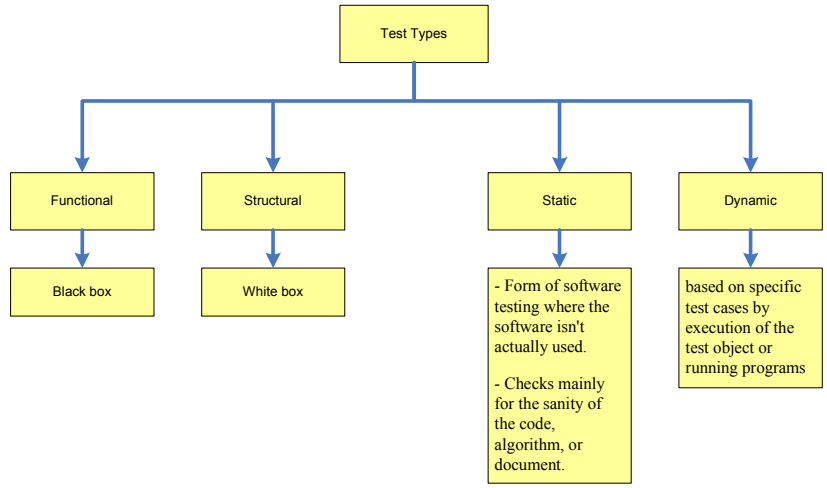
  d. Consequence

Answer: c

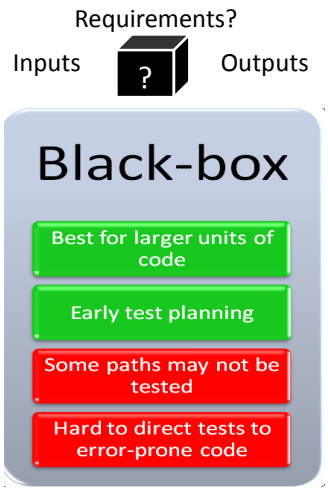
Test Strategy

- is the overarching approach to do testing
- is a road map for what testing will be done
- helps to answer questions such as
  - What types of testing needed? (manual or automated)
  - What is the actual test environment?
  - What are the testing resources and other resources needed?
  - When will testing occur?

Test types



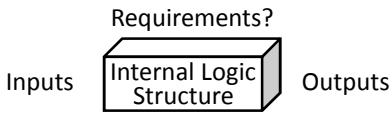
Test types – Black box

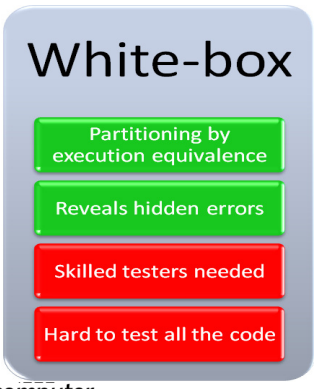


- Treats the software as a black box without understanding how the internal code is implemented
- Its purpose is to test the functionality according to the requirements
- Thorough test cases need to be provided to the tester, who then verifies that for a given input, the output value (or behavior) is the same as the expected value specified in the test case

Disadvantages
Advantages

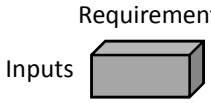
Test types – White box

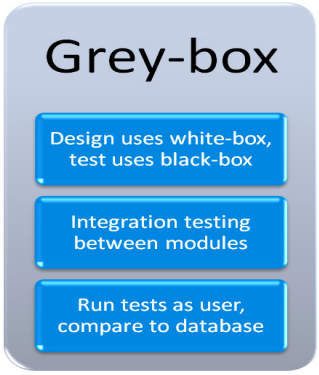




- Process that deals with the internal logic and structure of the code
- The test specifier uses knowledge of the internal structure of the software to derive test cases
- The test cases cannot be determined until the code has actually been written

Test types – Gray box





- Involves having access to internal data structures and algorithms for purposes of designing the test cases
- Testing is done at the user or black-box level

E.g.

Useful when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test

Discussion question

The most important aspect of structural testing (also known as "white-box" testing) is its ability to:

a. Reveal the presence of defects in various parts of the code
b. Establish the correctness of the module
c. Prove that every statement in the module is reachable
d. Prove that the module has a low cyclomatic complexity

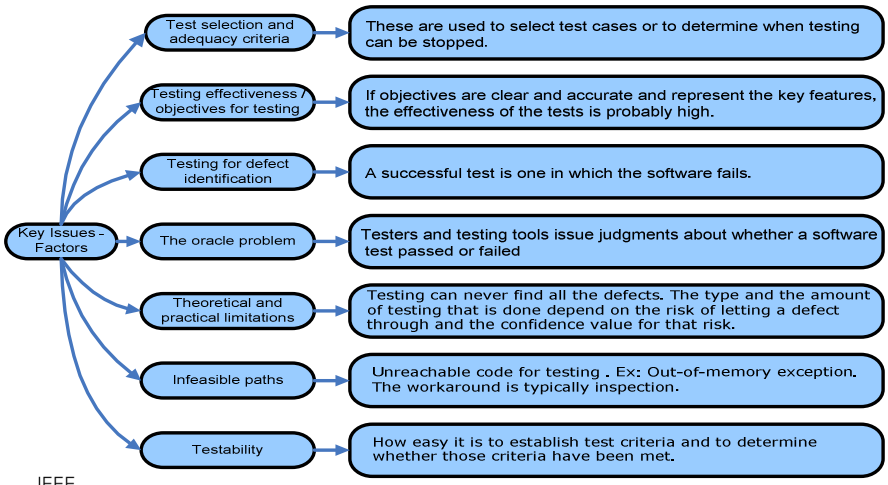
Answer: a

Key Issues – Decisions

When preparing a test plan, the tester must decide

- What items to test?
- At what level to test?
- What testing sequence to use?
- How the test strategy will be applied?
- What the test environment must include?

Key Issues – Factors



The Oracle problem

- An oracle is any (human or mechanical) agent which decides whether or not a program behaved correctly in a given test

A complete oracle would have three capabilities and would carry them out perfectly:

- A ***generator***, to provide predicted or expected results for each test.
- A ***comparator***, to compare predicted and obtained results.
- An ***evaluator***, to determine whether the comparison results are sufficiently close to be a pass.

Discussion question

A test oracle:

a. Generates predictions of expected test results.

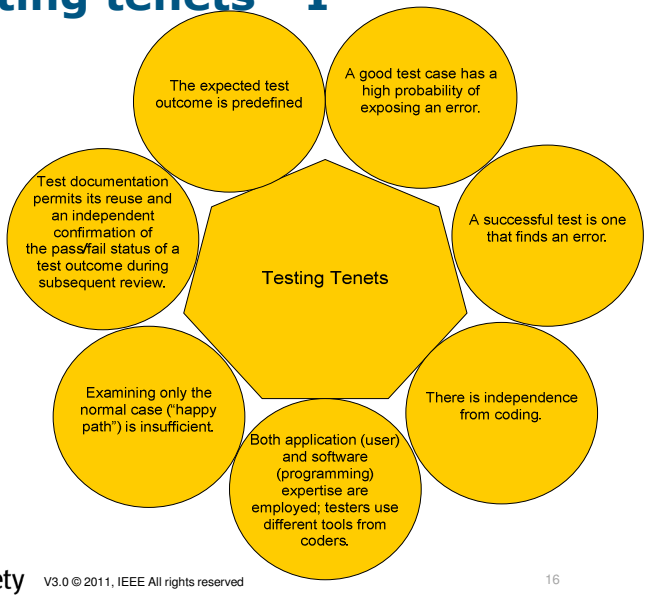
b. Manages the running of program tests.

c. Generates test data for the program to be tested.

d. Counts the number of times that a particular statement has been executed.

- Answer: a

Testing tenets - I



Discussion question

Which of the following is a testing tenets?

a. The expected test outcome is unknown until the test occurs.

b. A successful test is one that shows no errors.

c. Coding and testing are independent from each other.

d. Examining just the normal case is sufficient.

Answer: c

Relationship of Testing to other activities – I

- Testing is limited to the execution of the code, but there are other V&V techniques that are static.

E.g.

- Software reviews and inspections
- Code reading
- Correctness proofs and formal verification
- Symbolic execution
- Program proving
- Anomaly analysis

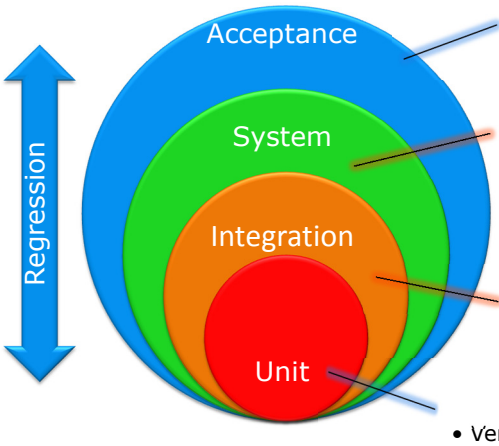
Relationship of Testing to other activities - II

- There are testing and test-related activities that are performed in other phases of the life cycle
  - **Requirements phase**: All requirements are checked for completeness, consistency, adherence to standards, etc.
  - **Design phase**: Documents, models, and prototypes are checked for inconsistencies
  - **Construction phase**: Code is reviewed by peers and technical experts and other tests are run

Content Area 2

Test Levels

Target of the test



- Done by Provider and customer
- Determines if system meets needs

- Assessment of whole system behavior
- Collecting information that will help direct the product release
- Discovering bugs that cannot be attributed to individual components

- Focuses on finding interface errors between units or components
- Focus is on the bugs that cannot be seen in unit testing

- Verifies proper functioning of that unit
- No interaction between units

Types of test - I

- Conformance or correctness testing
  - test cases are selected to verify that the observed behavior conforms to specifications
  - can be done at the unit, integration, and system levels
- Reliability achievement and evaluation testing
  - test cases are generated randomly
  - frequently used functionalities are sampled most densely

Types of test - II

- Regression testing
  - retesting to ensure that modifications previously passed tests still does
- Performance testing
  - aimed at verifying that the system meets performance requirements such as capacity and response time.

Discussion question

Which of the following are typically inputs to the system level testing process

I) System Performance Requirements
II) System Functional Requirements
III) Program Code
IV) System Design Specifications

a. I and II
b. I, II and IV
c. I
d. I, II, III and IV

Answer: a

Types of test - III

- Functional testing
  - software is tested for the functional requirements
- Volume testing
  - performance testing usually associated with batch processing
- Stress testing
  - Pushing performance to maximum design load and beyond to determine the limits of the system

Types of test - IV

- Load testing
  - performance testing associated with transaction-processing systems
- Back-to-back testing
  - same test is run on two implemented versions
- Configuration testing
  - When software is used in more than one configuration, it must be tested in each configuration

Types of test - V

- Usability testing
  - to evaluate the ease of use,
  - the ergonomics of the human-computer interface
  - effectiveness of the user documentation
- Recovery testing
  - to the software restart capabilities following a disaster or unanticipated shutdown
- Alpha testing
  - an independent test group simulates real-world use

Types of test - VI

- Beta testing
  - the software is distributed as an early release version to a representative group of users for testing
- Acceptance testing
  - to test customer requirements against system behavior
- Installation testing
  - done after all software and acceptance testing is complete

Discussion question

Beta testing is:

a. An inexpensive way to conduct a software quality assurance program

b. A reasonable way to check for software compatibility problems with various machine configurations

c. The best way to get focused user feedback to support user interface improvements

d. A guaranteed way to obtain positive product publicity

Answer: b

Types of test - VII

- Validation testing
  - ensures that the system meets the user's needs when operated in the user's environment
- Security testing
  - designed to find out how well the system can protect itself from unauthorized access, computer crime, or sabotage

Discussion question

Which of the following is MOST LIKELY performed outside of the organization?

a. Alpha testing

b. Beta testing

c. Stress testing

d. Usability testing

Answer: b

Content Area 3

Test techniques

Specification based testing techniques – I

- **Equivalence partitioning**
  - Reduces the number of test cases to a necessary minimum
  - Input domain is subdivided into a collection of subsets, or equivalent classes
  - Test cases are specified to include at least one value from each defined equivalence class.

Example: Function that passes a parameter "month" of a date.

Discussion question

Suppose you have a binary search routine that handles a special case list size of 13 elements. If the structure of the algorithm is used to identify equivalence classes, what is the minimum number of test cases required to test all equivalence class boundaries of this routine?

a. 2

b. 3

c. 4

d. 5

Answer: d

Specification based testing techniques – II

- **Boundary-value analysis**
  - Determines test cases that detect off-by-one errors
  - Test cases are selected based on the idea that errors are more likely to occur at the extreme values (boundary values)

Example: boundary test would verify that adding a month to a December date would produce a January result and increment the year

```
If (month > 12) then
  month = month – 12;
  year = year + 1;
end if
```


Specification based testing techniques – III

- **Decision tables**
  - used to build a complete set of test cases
  - table specifies a relationship between classes of conditions and actions
  - test cases are derived from each instance of a relationship.

		Rules							
Condition	**Values**	1	2	3	4	5	6	7	8
Condition 1	*Y, N*	Y	Y	Y	Y	N	N	N	N
Condition 2	*Y, N*	Y	Y	N	N	Y	Y	N	N
Condition 3	*Y, N*	Y	N	Y	N	Y	N	Y	N
Action									
Action 1		X			X				X
Action 2			X				X		X

Specification based testing techniques – IV

- **Cause–effect graphing**

Content Area 3: Test techniques

  - similar to the use of decision tables
  - The logical relationships of the inputs, processes, and outputs are shown in the form of a graph that maps a set of causes to a set of effects.
  - There may be intermediate nodes in between that combine inputs using logical operators such as AND and OR
  - Constraints may be added to the causes and effects

Specification based testing techniques – V

- **Finite–state machine based tests**

Content Area 3: Test techniques

  - Used to create behavioral models for functional testing
  - Excellent tool for software modeling, user/developer communication, and testing
  - Test cases are generated to exercise selected states and transitions among them

Specification based testing techniques – V

- **Formal methods**

Content Area 3: Test techniques

  - formal methods use rigorously specified mathematical models and proofs to help ensure correct behavior
  - Two formal approaches are the use of pre-conditions and post-conditions

Pre-condition	**Post-condition**
A condition that is required to be true before making a property request	A condition that is guaranteed to be true after a successful property request

(IEEE Std. 1320.2-1998, IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97)

Specification based testing techniques – VI

Content Area 3: Test techniques

- **Random testing**
  - Produces data without reference to the code or specification
  - Provides a mathematical basis for analysis and fast generation of test cases
  - No guarantee that there will be complete coverage of the program
  - Illegal or extreme values may not be included

Discussion question

Equivalence Partitioning is a testing technique used in the following:

a. White box testing
b. Black box testing
c. Stress testing
d. Usability testing

Answer: b

Code based testing techniques

Control-flow-based criteria

Path testing

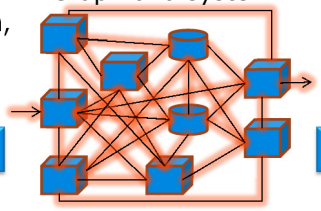
Statement, branch, and condition testing

Branch coverage

Executes every path at least once

No branching leads to abnormal behavior?

Graph of a system



Data-flow-based criteria

All definition-use paths

All definitions or all uses

Statement coverage

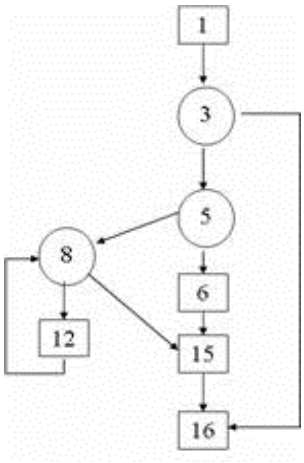
Executes all statements at least once

No side effects?

Discussion question

What is the Cyclomatic Complexity of the following flow graph for a program?

a. 3
b. 4
c. 9
d. 11



Answer: b

Discussion question

Consider the following code fragment:

```
input x;
if x > 0 then
      output x+1;
else
      output x-1;
end if;
while x > 5 then
      if x = 10 then
            output "blah"
      else
            output "ugh"
      end if;
      x = x – 1
end while;
```

The minimal number of test cases to achieve branch coverage of this program is:

a. 1
b. 2
c. 3
d. 4

Answer: b

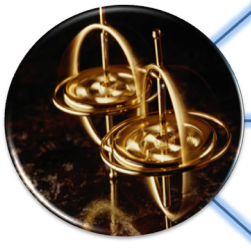
Discussion question

Suppose you have a software routine that controls a temperature sensor that drives a warning light on an airplane to notify the pilot of potential icing problems. The specification says that this light is to glow red whenever the temperature is strictly less than −20 degrees Celsius, yellow between −20 and 10 degrees Celsius (inclusive), and green for all temperatures strictly greater than 10 degrees Celsius. What is the minimum number of test cases needed to achieve statement coverage of this specification?

a. 2
b. 3
c. 6
d. 8

Answer: b

Fault-based Testing techniques



Error guessing

- Most plausible faults
- Historical information
- Experience

Fault seeding

- Single fault injected into each of several copies
- Run against correct version
- Fault tolerance?
- What happens in failure?
- Risk artificial faults may not be removed but it might be delivered to the customer

Mutation testing

- Test seldom-executed code
- Well-defined mutation operators
- Tests run against original and mutants
- All mutants should fail
- Syntactic faults may reveal more complex, real faults

Discussion question

All of the following dynamic testing techniques are based on program structure EXCEPT:

a. mutation analysis

b. random testing

c. dataflow testing

d. automatic path-based test data generation

Answer: b

Techniques Based on the Nature of the Application

OO testing	Many classes	Information hiding	Test program for each class
COTS	Black-box	Environment tested	Grey-box if data available
Web-based	Hardware variety	Dynamic content	Users control flow
GUI testing	Domain size	Sequences	Regression testing
Concurrent programs	Simultaneous testing	Less predictable failure modes	Probabilistic failures
Conformance testing	CTS	Discrete test cases	Atomic test cases
Real-time apps.	Time constraints	Timed tests	Pass, fail, inconclusive
Safety-critical systems	Physical component failure	Design error	Design diversity/ fault avoidance

Object oriented testing

- There may be a large number of independent classes that require testing
- Information hiding makes it difficult to determine whether the class is working correctly, because the state of internal data may not be accessible via the interface
- The programmer must develop an effective test program for each class
- The result is that the test program may well be more complex and larger than the class being tested

Commercial off-the shelf components

- There is increasing emphasis on the use of both commercial off-the-shelf and in-house components to help manage costs
- Components must be tested and adapted for use in each new environment
- Because these components cannot be modified, black-box techniques may be the most appropriate testing techniques

Web based applications

- There are several challenges in testing web-based applications
- The software needs to be tested on variety of hardware, network connections, operating systems, middleware, web server support, and web browsers
- Program and content may be generated dynamically
- Compatibility and interoperability are important qualities
- Users can control the flow simply by pushing the Refresh or Back buttons
- Web applications typically have faster maintenance requirements

Graphical user interface testing

- GUI testing helps ensures that the graphical user interface meets its specifications
- There are 3 significant challenges in testing GUIs.
  - **Domain size**: A GUI has many operations that need to be tested. A small program may have 200 to 300 possible GUI operations
  - **Sequences:** Some system functionality may require following some complex sequence of GUI events. This can become a serious issue when the tester is creating test cases manually
  - **Regression testing.** Regression testing is a challenge because a GUI may change across versions of the application, even though the underlying application may not.

Concurrent programs - I

- Testing concurrent programs poses two challenges
  - Testing requires that the tests themselves be concurrent programs
  - Failure modes of concurrent programs are less predictable and repeatable
  - Debugging or monitoring the program may introduce timing or synchronization artifacts that prevent the bug from appearing at all
  - Takes more time to design and execute concurrent tests than sequential ones

Concurrent programs - II

- When testing concurrent programs
  - Explore more of the state space
  - Explore more interleavings
  - Match thread count to the platform
  - Avoid introducing timing or synchronization artifacts

Conformance testing

- A typical conformance test suite (CTS) is constructed to cover the requirements of a complete specification or a significant portion of a specification
- The conformance test suites break down a protocol specification into discrete and atomic test cases
- Consist of several dozen to hundreds of test cases
- The test cases contain information on:
  - requirements for setting up the test case
  - What the inputs are
  - The expected response from the implementation under test

Real-time applications

- A typical conformance test suite (CTS) is constructed to cover the requirements of a complete specification or a significant portion of a specification
- The conformance test suites break down a protocol specification into discrete and atomic test cases
- Consist of several dozen to hundreds of test cases
- The test cases contain information on:
  - requirements for setting up the test case
  - What the inputs are
  - The expected response from the implementation under test

Safety critical systems - I

- Is a class of systems whose failure may cause injury or death to human beings
- Such systems have absolute demands regarding correctness of behavior
- Testing is designed to identify and eliminate possible bugs and to ensure correctness of behavior
- There are two major causes of failure in safety-critical systems
  - physical component failure and
  - design error

Safety critical systems - II

- Two approaches to increase the system's ability to function correctly are employed. They are
  - **Design diversity**
    - A technique is which multiple versions of the application are produced from the same specification
  - **Fault avoidance**
    - Uses formal methods in which models of the system are developed and analyzed mathematically

Discussion question

Given an Object class of 10 public methods and 3 private methods how should one proceed in testing the class?

Discussion question

Which of the following statements is true about unit testing of object-oriented systems?

a. Unit testing best takes place at the method level

b. Unit testing best takes place at the class level.

c. Unit testing is infeasible.

d. Unit testing focuses on testing individual attributes rather than methods.

Answer: b

Testing Based on SEs Intuition and Experience - I

Content Area 3: Test techniques

Ad hoc testing	Exploratory testing
▪ Relies on – Intuition – Skills and – Experience of tester ▪ Useful for identifying special tests that may not be captured during formal approaches ▪ Sometimes considered as type of exploratory testing	"What's the best test I can perform right now?" ▪ Refers to test that are defined in advance ▪ Effectiveness of test relies on – S/W Engineers knowledge – Experience of tester – Types of failures and faults – Risk

Discussion question

Content Area 3: Test techniques

In designing a set of test cases that execute all linearly independent paths in a program, which of the following statements is true?

a. The set of needed test cases can typically be determined by a simple visual inspection of the program.

b. Static analysis tools exist that can automatically generate the required test cases.

c. It is often necessary to instrument the code and dynamically determine during testing which paths have not yet been executed.

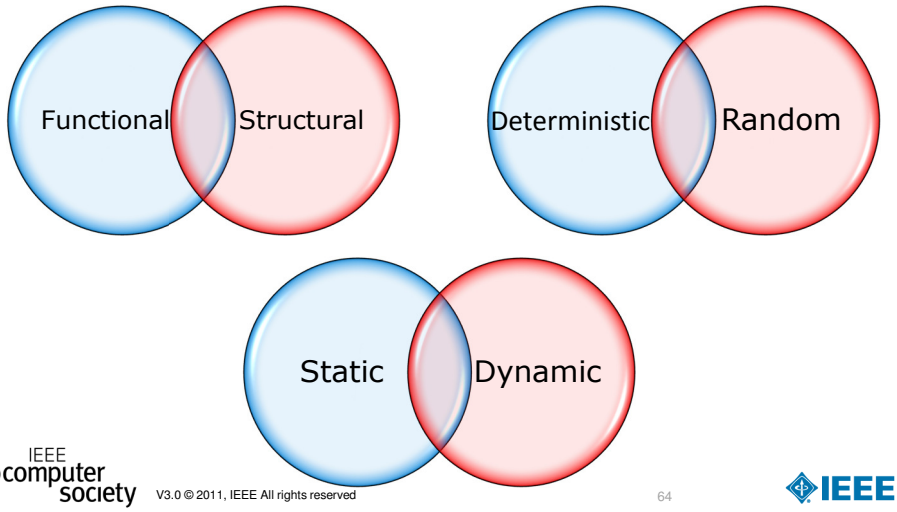
d. Special care must be taken to design test cases that execute dead code.

Answer: c

Testing Based on SEs Intuition and Experience - II

- Rapid feedback for new product or feature
- Learning product quickly
- Diversify testing after script-based testing
- Most important bug in least time
- Brief independent audit of other tester's work
- Investigating and isolating particular defect
- Status of particular risk to determine need for script

Selecting and Combining Techniques - I



Selecting and Combining Techniques - II

Content Area 3: Test techniques

- **Functional and structural**
  - When testing starts from the specifications, the functions are tested; this is functional testing
  - When test data is derived from the structure of the system, it is structural testing
- **Deterministic and random**
  - Test cases can be selected in a deterministic or randomly
- **Static and dynamic**
  - Static testing does not involve execution of the code. Dynamic analysis requires that the software be executed

Content Area 4

Human-Computer Interface Testing and Evaluation

Usefulness and usability - I

- Is it what the user really wants?
  - Usefulness: Meets user requirements
  - Usable: fitness for purpose, ease of use, ease of learning

Usefulness and usability - II

- **Efficiency**
  - refers to how efficiently the user performs a task using the system
  - How long does it take people to complete basic tasks?
    - E.g. Find something to buy, create a new account, and order the item from a site.
- **Learnability**
  - refers to how quickly users can learn to use the system for the first time.
  - How long does it take a new user to become productive?

Usefulness and usability - III

- **Accuracy**
  - refers to the errors made by the user when using the system, not the system errors
  - How many mistakes did people make? Were they fatal or recoverable?
- **Recall**
  - refers to how easy it is to remember how to use the system.
- **Emotional response**
  - most subjective attribute and refers to the user's subjective view of the system.

Heuristic evaluation

- Small set of evaluators examine the interface and judge its compliance with recognized usability principles
- The observers may answer the evaluators' questions to help them better assess the usability of the user interface
- Complementary to user testing; one should not substitute for the other

Cognitive walkthrough - I

- Similar to code walkthrough
- Evaluation done on attributes like
  - Efficiency
  - Learnability
  - Accuracy
  - Recall and emotional response

Focus is to

- to determine how easy a system is to learn and to see how users learn through exploration so that the system interface may be improved for ease of use.

Cognitive walkthrough - II

- A cognitive walkthrough requires the following items:
  - A detailed description of the prototype of the system.
  - A description of the task the user is to perform on the system.
  - A complete, written list of the actions needed to complete the task with the given prototype
  - Demographic information about the users, including their level of experience and knowledge of the processes

Discussion question

Which of the following is true of heuristic evaluations or cognitive walkthroughs?

a. Heuristic evaluation should not be a substitute for user testing.

b. In heuristic evaluation, observers should never offer hints on how users should proceed.

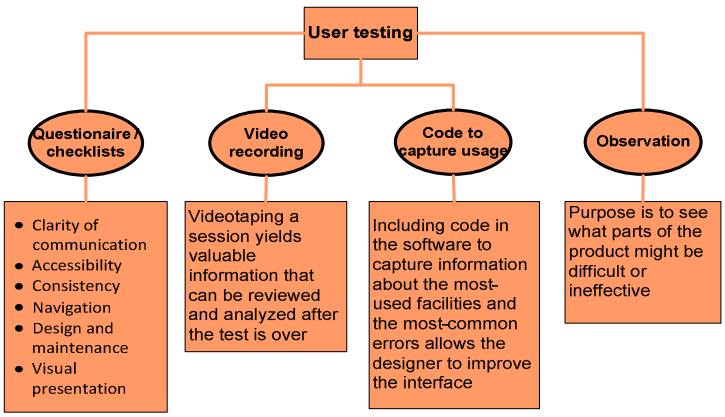
c. A cognitive walkthrough describes the prototype in detail and then has a user randomly select a task.

d. The focus of a cognitive walkthrough is to record the user's comments about the interface.

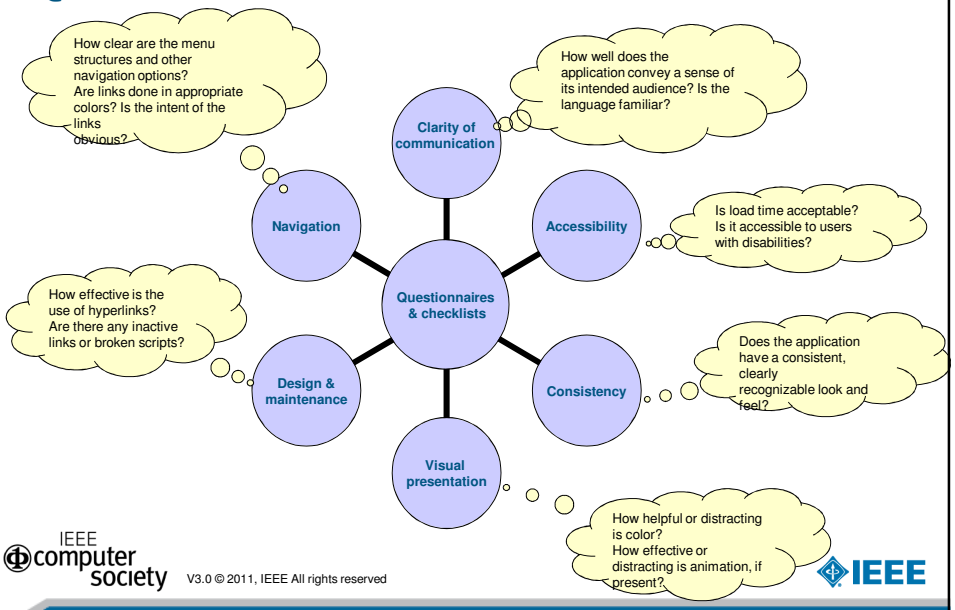
Answer: a

User testing

- User testing involves members of the target audience working through a set of tasks



Questionnaires and checklists



Observation

- The purpose of observing users is to see what parts of the product might be difficult or ineffective. Guidelines to make the user observation productive are
  - Limit the scope of the test to a specific aspect of the product
  - Be sure the tasks are real tasks that most users will do as part of their normal jobs
  - Ask participants to think aloud during the observation; this gives insight into their expectations for the product
  - Allow participants to work without any interference or extra help

Web usability and Feedback

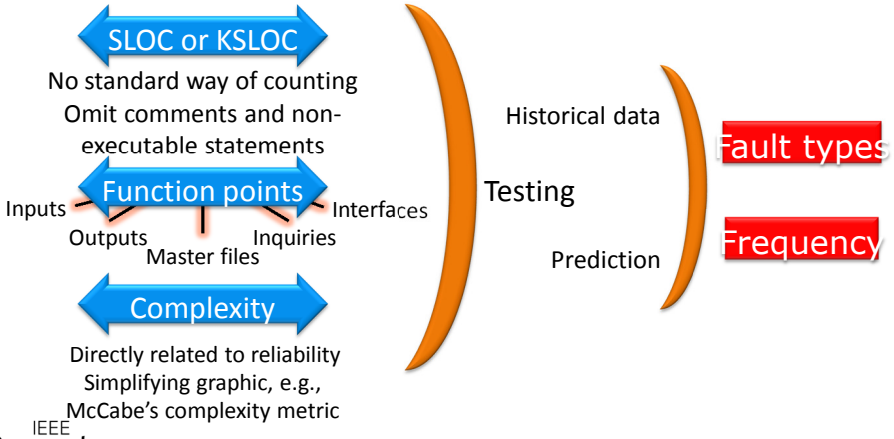
- Web usability guidelines
  - Have users perform real tasks and solve real problems
  - Use appropriate automatic evaluation methods for slow pages, missing links, jargon, accessibility, etc
- Feedback
  - Mechanism for submitting responses to developers
  - Empowering
  - Provide good information to the designer and coders to use to redesign and structure the interface

Content Area 5

Test – Related measures

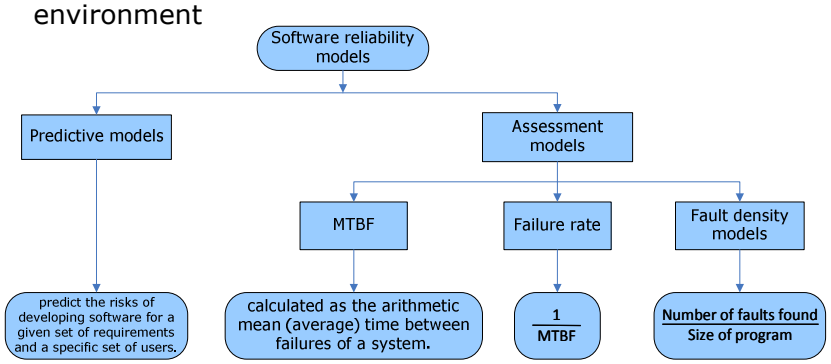
Program measurements

Includes program size, complexity, or structure may be used to guide testing



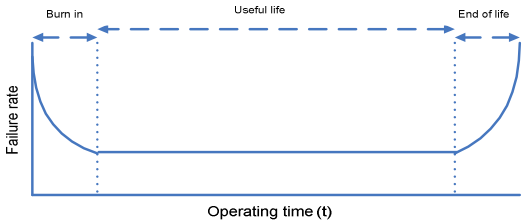
Evaluation of Program: Reliability Evaluation - I

- Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment

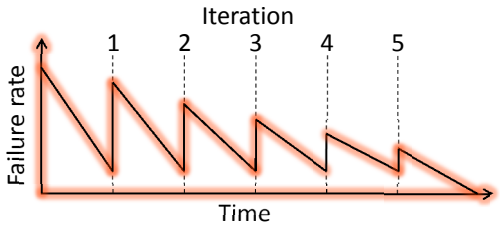


Evaluation of Program: Reliability Evaluation - II

- Component Failure Rate Curve



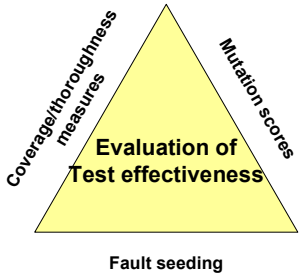
- Software Upgrade Failure Rate Curve



Evaluation of tests performed - I

- Mutation scores
  - Faults are introduced into several mutants of the program
  - Test cases are run on the original program and all mutant variations
  - One mutant has only one fault
  - **Mutation score** = 100 * D /(N – E)

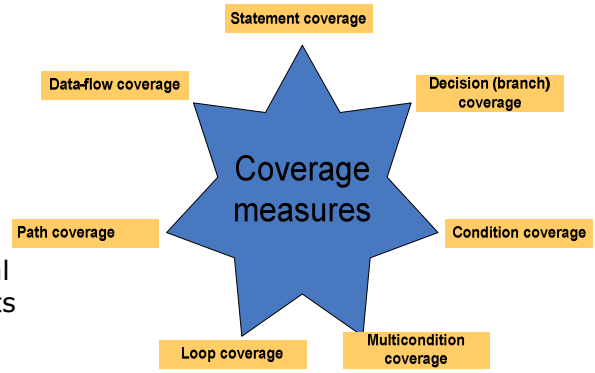
    Where D = dead mutants
    N = total number of mutants
    E = no. of equivalent mutants



- Fault seeding
  - Faults are introduced into the program before testing.
  - When tests are executed, these should be revealed, along with other faults

Evaluation of tests performed - II

- Coverage measures
  - Many tests cover a selected set of elements
  - measures the ratio of the elements covered to the total number of elements
  - are a way of estimating fault and reliability



Coverage metrics - I

- **Statement coverage**
  - requires sufficient test cases for each program statement to be executed at least once
- **Decision (branch) coverage**
  - requires sufficient test cases for each program decision or branch to be executed so that each possible outcome occurs at least once
- **Condition coverage**
  - requires sufficient test cases for each condition in a program decision to take on all possible outcomes at least once

Coverage metrics - II

- **Multi-condition coverage**
  - requires sufficient test cases to exercise all possible combinations of conditions in a program decision
- **Loop coverage**
  - requires sufficient test cases for all program loops to be executed for many iterations covering initialization, typical running, and termination (boundary) conditions
- **Path coverage**
  - requires sufficient test cases for each feasible path, basis path, etc., from start to exit of a defined program segment to be executed at least once
- **Data-flow coverage**
  - requires sufficient test cases for each feasible data flow to be executed at least once

Discussion question

Due to the large number of possible ways to get through a software program, which of the following is generally not feasible and is therefore usually scaled back to a level based on the risk or criticality of the software being tested?

a. Decision (branch coverage)

b. Exhaustive testing

c. Fault seeding

d. Mutation scores

Answer: b

Content Area 6

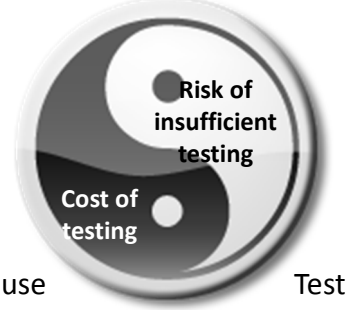
Test process

Practical considerations

Foster an attitude of cooperation and collaboration

- Attitudes/ego-less programming
- Test guides
- Test process management
- Test documentation and work products
- Internal vs. independent test team

Estimation and measurement

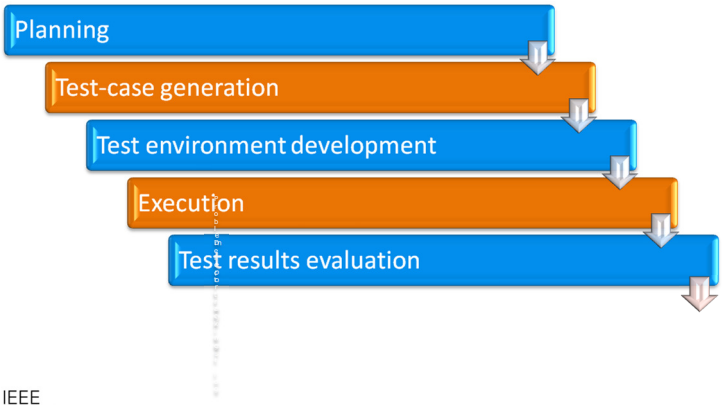


Test reuse and test patterns

Test termination

Test activities

Test activities start at the beginning of the software life cycle and continue throughout its stages

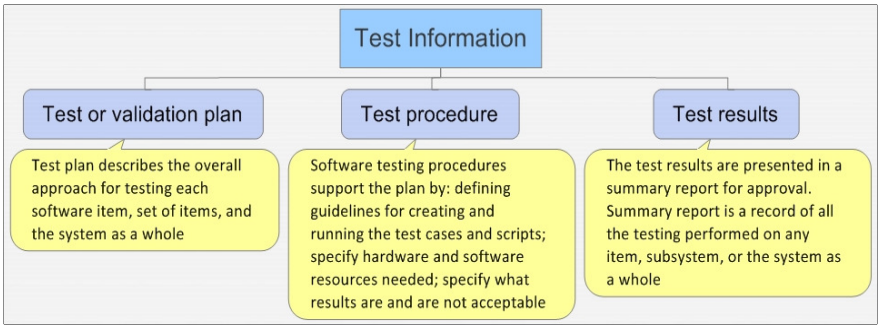


Content Area 7

Product documentation

Test Information Categories

Test information should be captured in these three categories irrespective of the software development model



ref 1-124

Test Plan - I

- Test plan describes the overall approach for testing
  - should state what items to be tested, the level at which they will be tested, the sequence in which they will be tested, and the strategies that will be applied
- Test plan documentation
  - May be started during the requirements phase as the verification criteria are established for each requirement
  - It is added to during design and construction

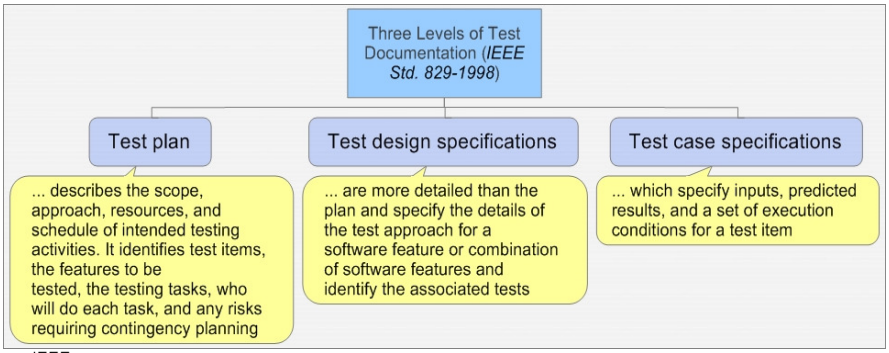
Contents for a test or validation plan (IEEE Std. 12207.1-1997)

a) Generic plan information
b) Test levels
c) Test classes
d) General test conditions
e) Test progression
f) Data recording, reduction, and analysis
g) Test coverage (breadth and depth) or other methods for assuring sufficiency of testing
h) Planned tests, including items and their identifiers
i) Test schedules
j) Requirements traceability
k) Qualification testing environment, site, personnel, and participating organizations

ref 1-125

Test plan - II

- The test plan documentation start at beginning
  - e.g., during the requirements phase as verification criteria for each requirement



Test Procedure

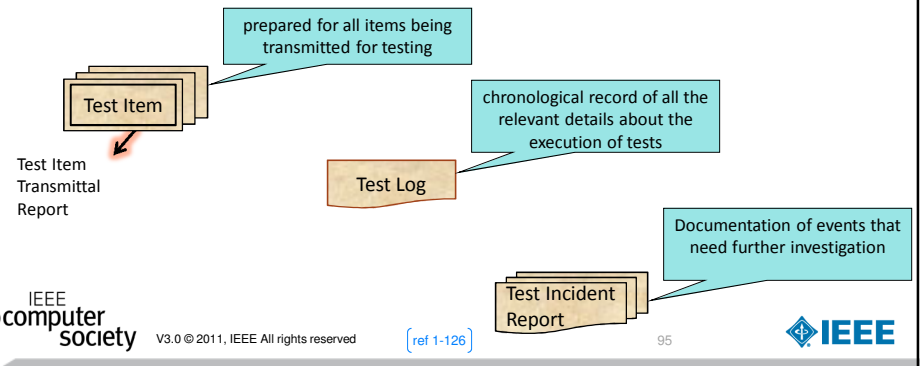
- Software testing procedures support the test plan
  - They define the guidelines for creating and running the test cases and scripts
  - They specify the hardware and software resources that are needed
  - They specify what results are and are not acceptable

Contents for a test procedure
(IEEE Std. 12207.1-1997)

a) Generic procedure information
b) Identification of test author
c) Identification of test configuration
d) Test objectives, requirements, and rationale
e) Test preparations (h/w, s/w, other) for each test
f) Test descriptions including
   1) Test identifier
   2) Requirements addressed
   3) Prerequisite conditions
   4) Test input
   5) Expected test results
   6) Criteria for evaluating results
   7) Instructions for conducting procedure
g) Requirements traceability
h) Rationale for decisions

Test Procedure: Documents

- Many documents created during the actual test activities
  - test transmittal reports, test logs, test incident reports etc.
  - Format of such reports should be prepared during construction phase
    - so that they can be completed quickly as needed



Test Results

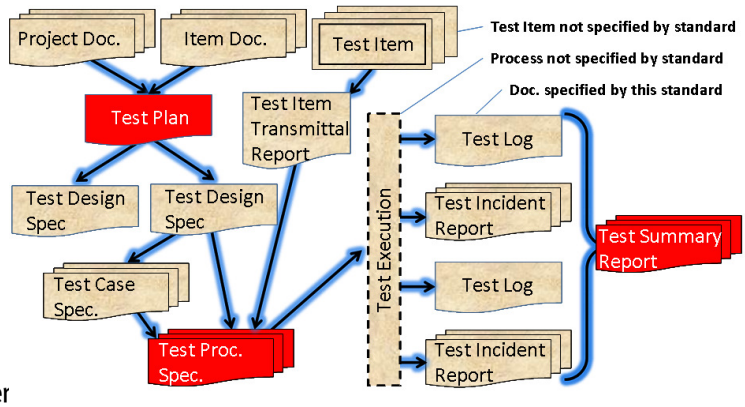
- The test results are presented in a summary report for approval
  - The summary report is a record of all the testing performed on any item, subsystem, or the system as a whole
- Most results will be reported during testing
  - But some results may be reported during construction
  - The format and contents of the summary report can be created during construction so that they are available as needed

Contents for a test results summary (IEEE Std. 12207.1-1997)

a) Generic report information
b) System identification and overview
c) Overview of test results, including
  1) Overall assessment of the software tested
  2) Impact of test environment
d) Detailed test results, including
  1) Test identifier
  2) Test summary
  3) Problems encountered
  4) Deviations from test cases/ procedures
e) Test log
f) Rationale for decisions

Relationship of Documentation Items

- **Figure shows relationship between elements of software test documentation as in *IEEE Std. 829-1998 (IEEE Standard for Test Documentation)***



 [ref 1-127]

Discussion Question

When should test cases for a unit test be determined?

How should they be determined?

References

ANSI/IEEE Std. 1008-1997, IEEE Standard for Software Unit Testing. New York: Institute of Electrical and Electronics Engineers, 1997

Binder, Robert. Testing Object-Oriented Systems: Models, Patterns, and Tools. Boston: Addison-Wesley, 2000.

Bizer, Boris. Black-Box Testing: Techniques for Functional Testing of Software and Systems. New York: John Wiley, 1995.

Guide to the Software Engineering Body of Knowledge (SWEBOK), 2004 ed. Los Alamitos, California: IEEE Computer Society Press, 2004.

IEEE Std. 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology. New York: Institute of Electrical and Electronics Engineers, 1990.

IEEE Std. 829-1998, IEEE Standard for Test Documentation. New York: Institute of Electrical and Electronics Engineers, 1998.