Test Techniques

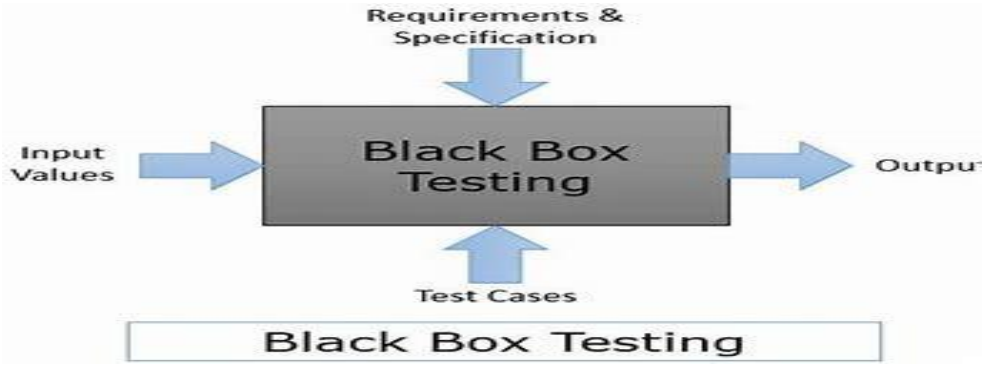
JAN, 2026

expleo

3 Testing Techniques

1.Black-box test techniques

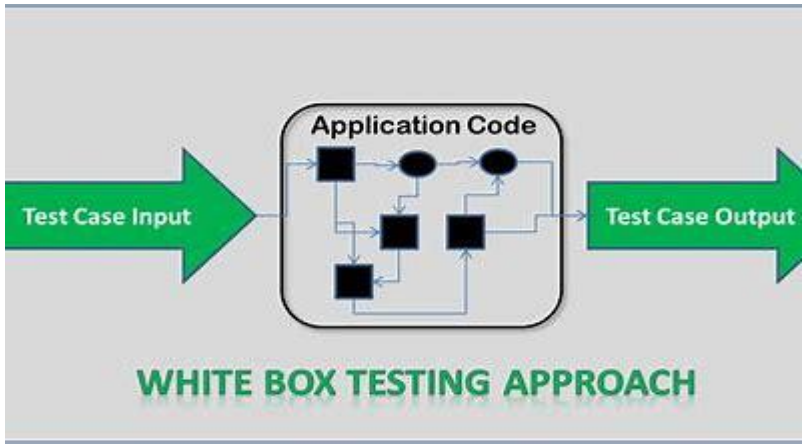
- also called behavioral or specification-based techniques : based on an analysis of the appropriate test basis, for example: requirements specification, use cases, user stories, business processes, or even the customer knowledge or common sense. Black-box test techniques concentrate on the behavior of the system under test, that is, on the inputs and outputs of the test object, without reference to its internal structure.



3 Testing Techniques

2. White-box test techniques

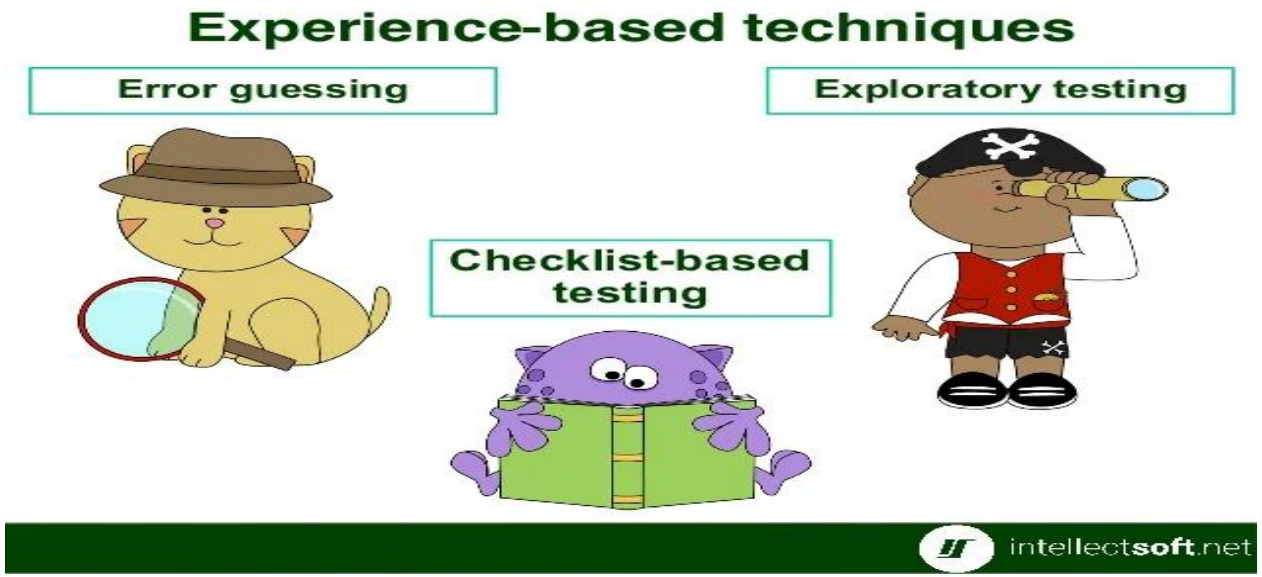
- also called structural or structure-based techniques : based on an analysis of the architecture, design, internal structure, or the code of the test object. They concentrate on the processing within the test object. The analysis of the internal structure of the test object is used to design test cases.



3 Testing Techniques

3. Experience-based test techniques

- utilize the knowledge and experience of developers, testers, and users to determine what should be tested. These techniques are often combined with black-box and white-box test techniques.



3 Testing Techniques

TEST TECHNIQUES (in the Foundation Syllabus)		
BLACK-BOX TEST TECHNIQUES	WHITE-BOX TEST TECHNIQUES	EXPERIENCE-BASED TEST TECHNIQUES
Equivalence Partitioning	Statement Testing	Error Guessing
Boundary Value Analysis	Decision Testing	Exploratory Testing
Decision Table Testing		Checklist-based Testing
State Transition Testing		
Use Case Testing		

Black-box Testing Techniques

- The common characteristics of the black-box techniques, according to the syllabus, are as follows:

- Test conditions, test cases, and test data are derived from a test basis that may include software requirements, specifications, use cases, and user stories.
- Test cases may be used to detect gaps between the requirements and the implementation of the requirements, as well as deviations from the requirements.
- Coverage is measured based on the items tested in the test basis and the technique applied to the test basis.

Black-box Testing Techniques

Table 4.1 Summary of the black-box techniques in the Foundation syllabus

Technique	Model	Typical defects that can be found using this technique
Equivalence partitioning	Domain model	Wrong handling of data/domain values
Boundary value analysis	Domain model	Wrong handling of data on the domain boundaries
Decision table testing	Decision logic model	Wrong handling of the business rules
State transition testing	Behavioral model	Wrong handling of transitions between states
Use case testing	Business process model	Wrong handling of the business processing

Black-box Testing Techniques – Equivalence Partitioning

-Equivalence partitioning (EP) is a domain-oriented technique. Given a domain of some variable, a tester divides it into a finite number of nonempty subsets called equivalence classes (or equivalence partitions, or just partitions).

-The idea is that the partitioning should be done in a way that for any two values from the same equivalence class, the system under test behaves in a similar way.

-The EP technique tries—in some sense—to overcome one of the fundamental principles of testing, saying that "exhaustive testing is impossible": from the potentially infinite number of possible inputs, outputs, or internal variables, we derive a finite number of test cases.

Black-box Testing Techniques – Equivalence Partitioning

-Here are some important considerations about equivalence partitioning:

- It can be used to any nonempty domain—it does not need to be a numerical domain, it may be any ordered or unordered set of values, finite or infinite, discrete or continuous, for example: {2, 4, f, d, aa}, an interval [0, 1], an infinite set {0, 1, 2, ...} of nonnegative integers etc.
- The proper partitioning requires that each domain element belongs to exactly one partition and that all partitions are nonempty.

Black-box Testing Techniques – Equivalence Partitioning

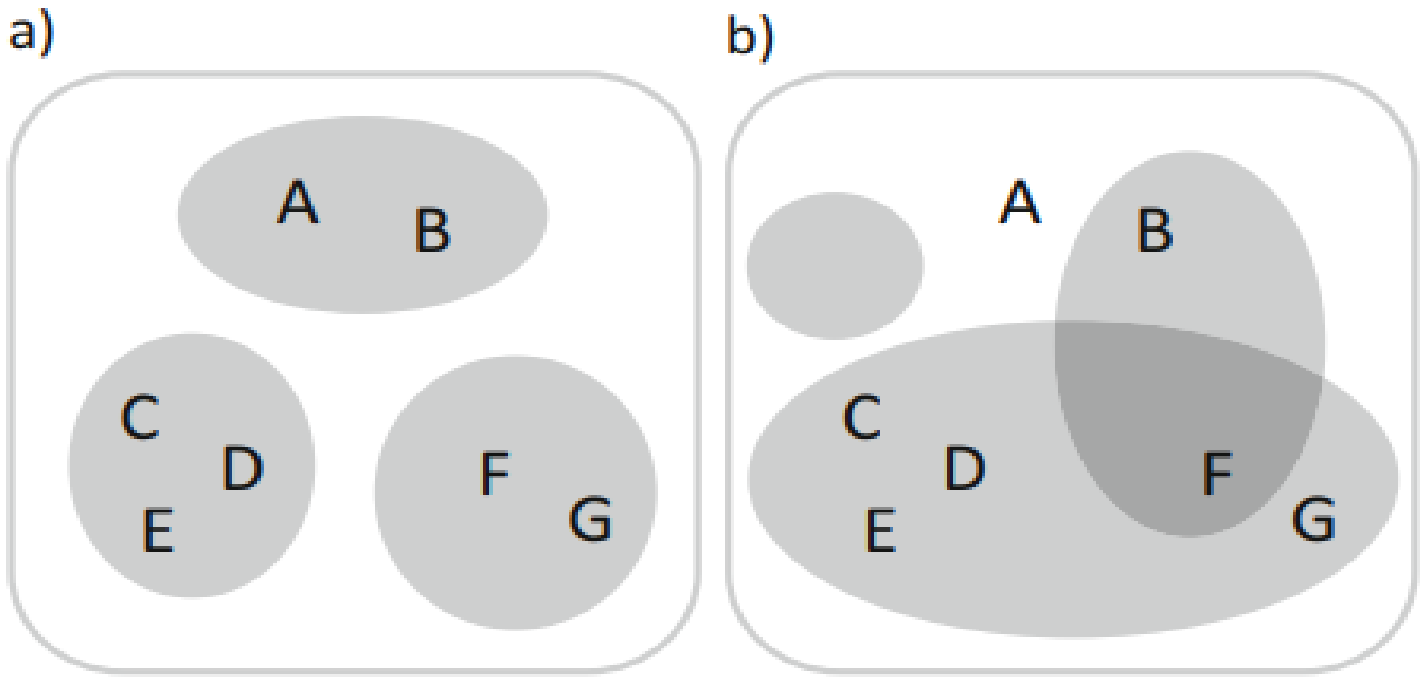
- Equivalence partitioning can be performed not only for the input variables, but also for output ones or any other kind of variables — each domain can be subject to this technique. There are two types of partitions: valid and invalid. Valid partitions represent correct, expected values, and incorrect partitions represent incorrect, unexpected values.

- The coverage is computed as the number of different classes from which we picked the elements and tested them, divided by the number of all partitions.

- Taking more than one element from one partition to the test suite does not increase the coverage.

Black-box Testing Techniques – Equivalence Partitioning

Fig. 4.1 An example of good and wrong equivalence partitioning. **(a)** Correct partitioning, **(b)** incorrect partitioning



Black-box Testing Techniques – Equivalence Partitioning

-The coverage is calculated as the number of class from which we selected the elements to tests, divided by the total number of all identified equivalence classes.

-For example, for a partitioning p1 = {1, 2, 3}, p2 = {4, 5, 6}, p3 = {7, 8}, and a set {1, 4} of test input values, the coverage is 2/3 66%, because we covered only p1 and p2.

-100 % Coverage ? Give Test Cases

Black-box Testing Techniques – Boundary Value Analysis

-Boundary value analysis (BVA) is, like EP, also a domain-oriented technique. It is built upon the equivalence partitioning method.

-The first step in applying the BVA is to perform the equivalence partitioning of a given domain. The difference between EP and BVA lies in the way in which we choose class representants to tests.

-In the case of EP, we had to take one arbitrary element from each class. In the case of BVA, we take only the so-called boundary values of the identified classes.

-The idea of this method is that the defects are often found on the boundary values of the domains.

Black-box Testing Techniques – Boundary Value Analysis

-Once we have a proper partitioning, we need to identify the boundary values and select the test values corresponding to them. There are two variants of the BVA method: 2-value and 3-value.

-In a 2-value version, for each identified equivalence class, the test values are:

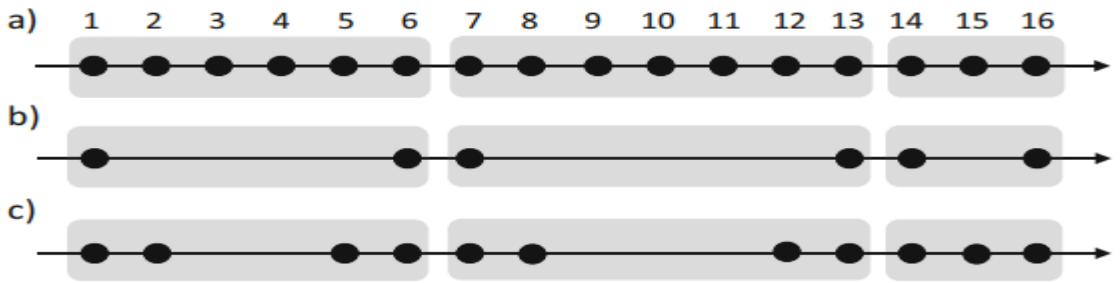
- The boundary values of this class (i.e., the minimal and maximal elements)
- The element just below the minimal one
- The element just above the maximal one

Black-box Testing Techniques – Boundary Value Analysis

-In a 3-value version, for each identified equivalence class, the test values are:

- The boundary values of this class (i.e., the minimal and maximal elements)
- The elements just below and just above the minimal one
- The elements just below and just above the maximal one

-Suppose we have a domain {1, 2, ..., 15, 16}, which is split into three valid equivalence classes: {1, ..., 6}, {7, ..., 13}, and {14, ..., 16} The



Black-box Testing Techniques – Decision Tables

-Decision tables are the models used in the test deign when we want to test the business logic. Each column of the decision table represents a single decision rule in a form:

IF (conditions) THEN (actions)

-The process of deriving the test cases from the decision tables is as follows:

1. Identify all possible conditions.

2. Identify all the corresponding actions that may occur in the system.

3. Generate all possible combinations of conditions; each single combination forms a separate column in the decision table

Black-box Testing Techniques – Decision Tables

4. For each combination of conditions, identify which actions should hold and which should not; fill the corresponding fields in a given column below the corresponding combination of conditions.

5. For each column design a test case in which a given combination of conditions hold; the test should verify if the corresponding actions hold.

Black-box Testing Techniques – Decision Tables

-Let us see on a practical example how the decision tables work. Suppose we have a system that checks if a candidate for a driving license fulfills all requirements to receive the license. The requirement specification describes the below mentioned business rules as follows:

> The operator enters the following information to the system:
>
> - The result of the theory exam (an integer value in the range 0–100 points)
> - The result of the practice exam (an integer value greater or equal to zero, representing the number of errors done by the candidate during the practice exam)
>
> A candidate receives the driving license, if she gained at least 85 marks (out of 100) from the theory exam and if she made at most 2 errors during the practice exam. In case of failing on one of these criteria, the candidate is eligible to retake the exam. In case of failing in both parts, the candidate is required to retake the driving lessons.

Black-box Testing Techniques – Decision Tables

-Let us go through the process described above to derive test cases for this problem.

1.Identify all possible conditions. The possible conditions on which the system's decisions depend are obviously:

- The result of the theory exam (number of points)
- The result of the practice exam (number of errors)

- The decisions will depend on the results of these exams. From the specification we know that the theory exam is passed when a candidate receives at least 85 points. The practice exam is passed, when the number of mistakes is 2 or less.

Black-box Testing Techniques – Decision Tables

2. Identify all the corresponding actions that may occur in the system. The system may take the following decisions:

- Should a candidate receive the driving license? (possible values: YES or NO)
- Should a candidate retake the theory exam? (possible values: YES or NO)
- Should a candidate retake the practice exam? (possible values: YES or NO)
- Should a candidate retake the driving lessons? (possible values: YES or NO)

Black-box Testing Techniques – Decision Tables

3. Generate all possible combinations of conditions; each single combination forms a separate column in the decision table. Each condition has two possible values, so we have 2 x 2 = 4 possible combinations of conditions (and, hence, four columns in the decision table):

4. For each combination of conditions identify which actions should hold and which should not; fill the corresponding fields in the given column below the corresponding combination of conditions. Now we go through the columns one by one and for each of them, we identify the expected behavior of the system:

Black-box Testing Techniques – Decision Tables

5. For each column, design a test case in which a given combination of conditions holds; the test should verify if the corresponding actions hold.

Table 4.3 The complete decision table for the driving license problem

CONDITIONS ↓ RULES →	1	2	3	4
Points ≥ 85?	YES	YES	NO	NO
Errors ≤2?	YES	NO	YES	NO
ACTIONS ↓				
Grant the license?	YES	NO	NO	NO
Retake theory?	NO	NO	YES	YES
Retake practice?	NO	YES	NO	YES
Retake driving lessons?	NO	NO	NO	YES

Black-box Testing Techniques – State Transition Testing

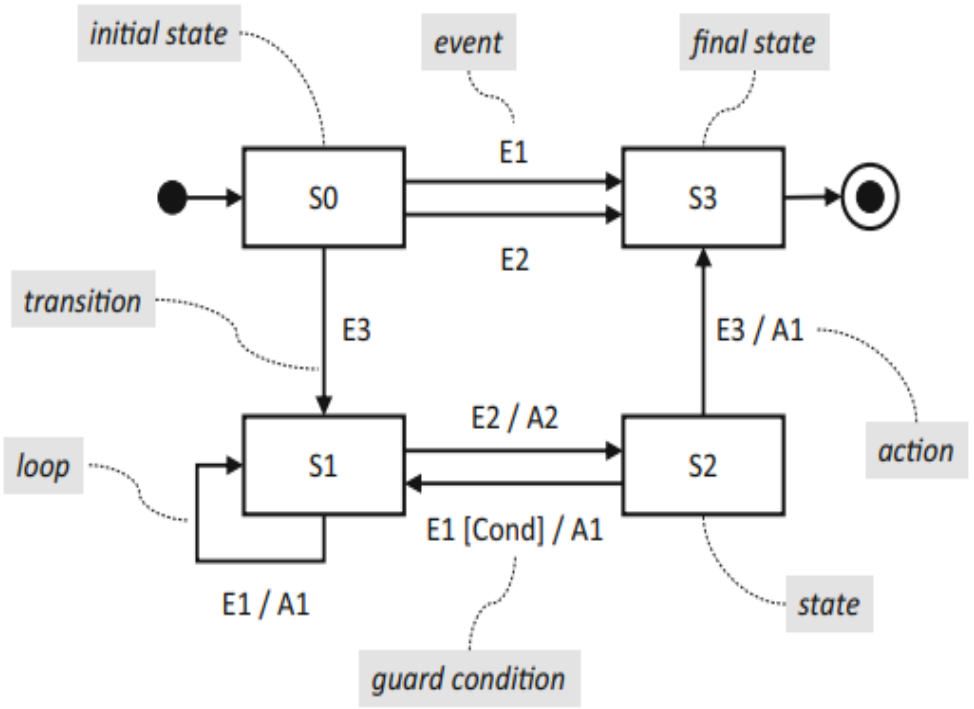
-State transition testing is used when we want to check the behavioral aspect of the system, that is—its behavior in time. The model for such tests is a finite state machine, called a state transition diagram.

-The state transition diagram consists of:

- States—they represent the possible conditions in which the system can be.
- Transitions—they represent the possible machine changes from one state to another.
- Events—they represent the external things that may happen and trigger the state change.

Black-box Testing Techniques – State Transition Testing

- Actions—they represent the possible activities done by the machine when changing the state.
- Guard conditions—they represent the additional conditions for the transitions that need to be fulfilled in order to trigger the transition; in particular, they are used if there is more than one possible transition from the same state under the same event.



An example of a state transition diagram

Black-box Testing Techniques – State Transition Testing

-Transition (0-Switch) Coverage In order to achieve the coverage, we need first to identify all the 0-switches in the diagram, that is—all the valid transitions. These are

(1) S0 -> (E1) -> S3

(2) S0 -> (E2) -> S3

(3) S0 -> (E3) -> S1

(4) S1 -> (E1) -> S1

(5) S1 -> (E2) -> S2

(6) S2 -> (E1) [Cond] -> S1

(7) S2 -> (E3) -> S3

Table 4.9 Minimal set of test cases for achieving the 0-switch coverage

#	Test	0-switches covered
1	S0 → (E1) → S3	(1)
2	S0 → (E2) → S3	(2)
3	S0 → (E3) → S1 → (E1) → S1 → (E2) → S2 → (E1) [Cond] → S1 → (E2) → S2 → (E3) → S3	(3), (4), (5), (6), (7), (8)

Black-box Testing Techniques – Use Case Testing

-A use case specifies the behavior of a system that interacts with one or more actors resulting in an observable result of value to the actors.

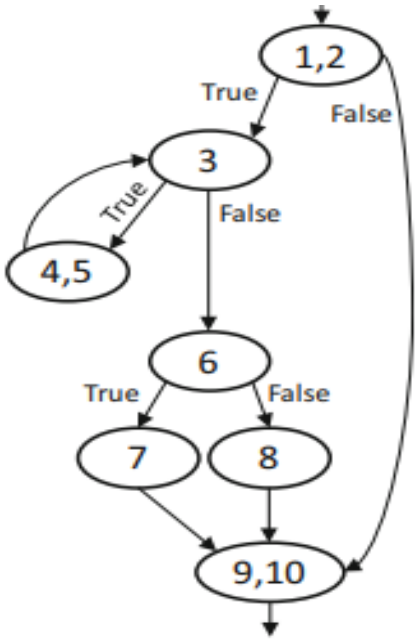
-An actor is typically a user, but it might be as well the other system. Well-structured use cases denote essential system or subsystem behaviors only, and are neither overly general nor too specific.

White-box Testing Techniques – Statement Testing and Coverage

-Statement testing coverage requires that each executable statement is exercised at least once by at least one test. The coverage is defined as the number of executable statements exercised divided by the total number of the executable statements.

-Let us look at the CFG of the 10-line code from Fig.

```
1.  INPUT x, y
2.  IF (x>0) THEN
3.    WHILE (x>0) DO
4.      y := y+1
5.      x := x-1
6.    IF (y>0) THEN
7.      y := 0
      ELSE
8.      y := 1
9.  x := y
10. RETURN x+y
```



White-box Testing Techniques – Statement Testing and Coverage

-The minimal number of test cases that achieve the (100%) statement coverage is 2, because lines 7 and 8 cannot be executed within one test.

-So the two sample test cases may, for example, go along the following paths:

- Test 1: 1, 2, 3, 4, 5, 3, 6, 7, 9, 10
- Test 2: 1, 2, 3, 6, 8, 9, 10

-If our test suite contains only the first test, this suite would achieve 90% statement coverage, as the test exercises 9 out of 10 executable statements.

-In case of the test 2, this would be 70%.

-These two tests together exercise all the executable statements, so together, they achieve the 100% statement coverage.

White-box Testing Techniques – Decision Testing and Coverage

-Decision coverage requires that every decision outcome (both TRUE and FALSE) should be exercised at least once by at least one of our tests.

-For example, in the code from the previous paragraph, we have three decision points—these are the nodes: (1, 2), (3), and (6).

-Each of them has two outgoing transitions; hence altogether, we have six test conditions to cover.

-The minimal number of test cases achieving the decision coverage in this example is 3.

-For example,

- Test 1: 1, 2, 9, 10 (covers the test condition 2)
- Test 2: 1, 2, 3, 4, 5, 3, 6, 7, 9, 10 (covers the test conditions 1, 3, 4, and 5)
- Test 3: 1, 2, 3, 6, 8, 9, 10 (covers the test conditions 1, 4, and 6)