6. Software testing preparation. “Black-box” and ”White-box” testing techniques

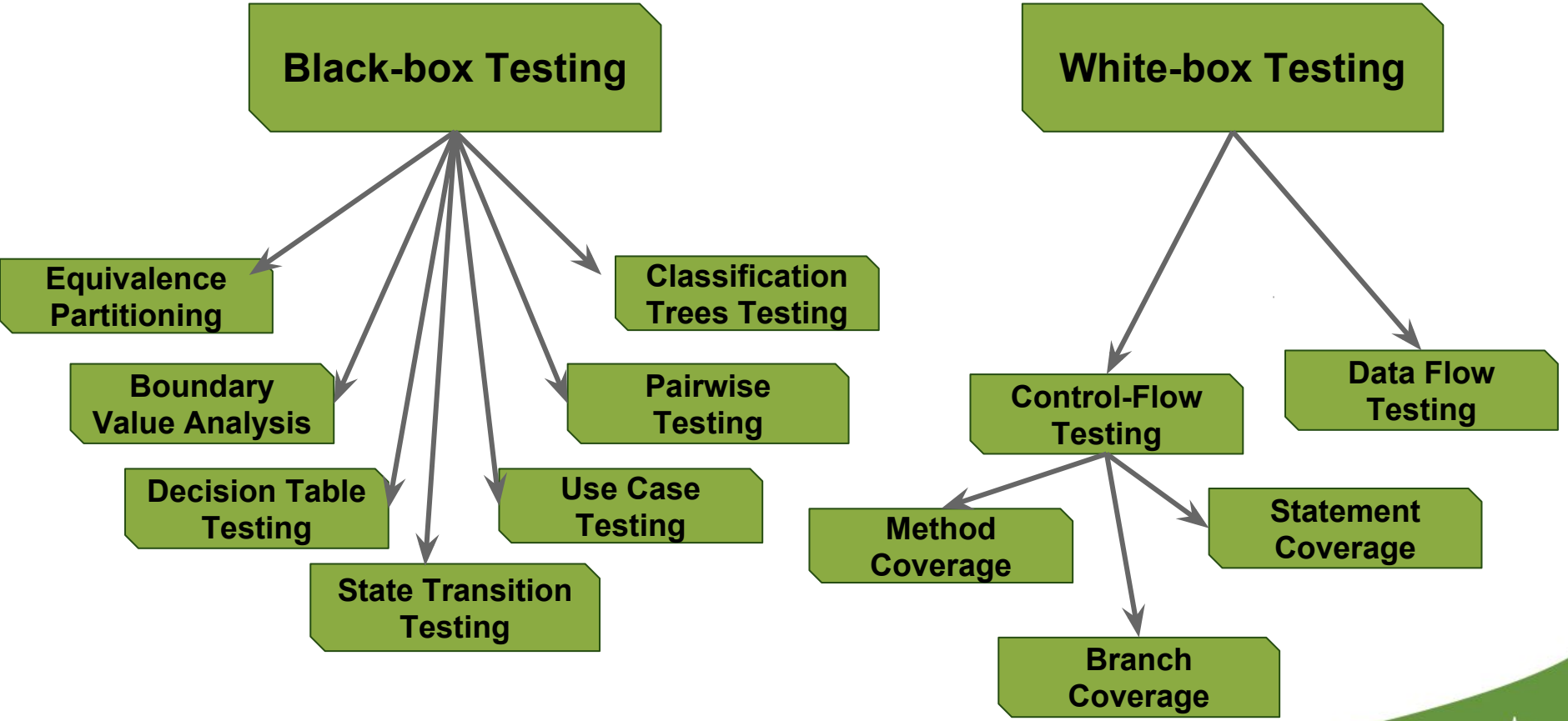
How to prepare tests?

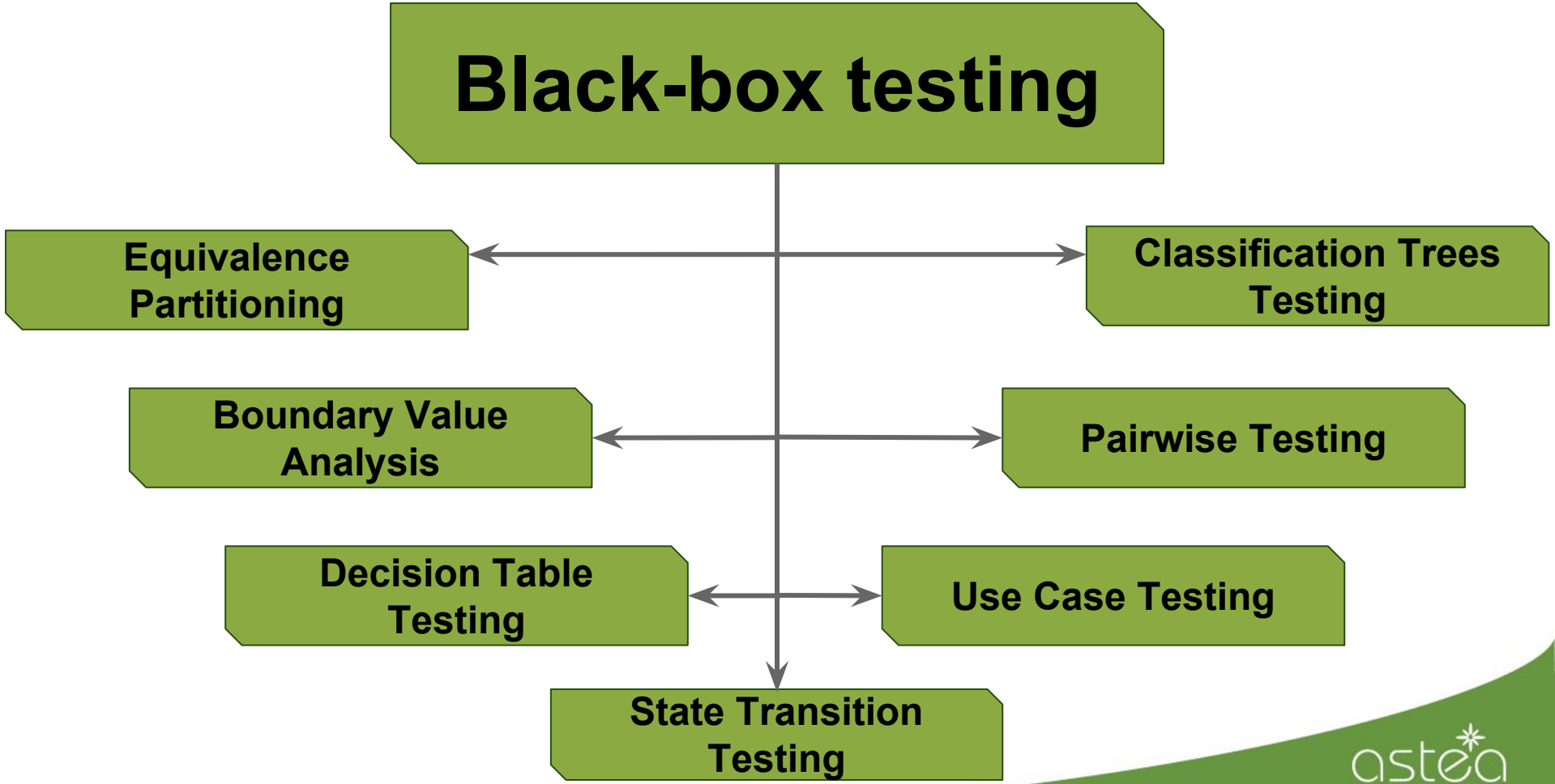
Astea Solutions QA Team

astea SOLUTIONS

Questions

- List the types of testing based on:
  - object of testing
  - knowing internal structure
  - time of test execution
  - positivism of test scenarios
  - degree of isolation
  - degree of automation
  - level of preparation
- What is static testing?



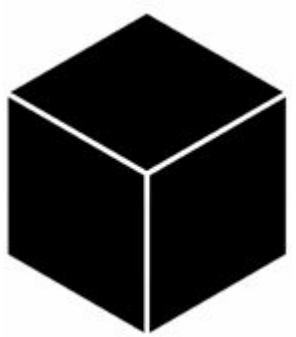


Black-box testing

Black-box testing techniques

Black-box techniques are a way to derive and select test conditions, test cases, or test data

- Based on an analysis of the test basis documentation
- Also called specification-based or behavioral techniques
- Tests are based on the way the system is supposed to work
-



Black-box techniques are divided into two main subtypes

- Functional
  - What the system does?
- Non functional
  - How the system does what it does?

Black-box testing techniques (2)

The choice of test techniques to be used depends on a number of factors:

- Type of the system
- Customer requirements
- Level of risk
- Documentation available
- Knowledge of the testers
- Time and budget

Equivalence Partitioning

Equivalence Partitioning

- Divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same (i.e. the system should handle them equivalently)
- Equivalence partitions are also known as equivalence classes
- We need to test only one condition from each partition

Equivalence Partitioning (2)

Example:

A field for age is tested

- if age is between **0-3 - baby**
- if age is between **4-12 - kid**
- if age is between **13-19 - teenager**
- etc...

Tests:

- Group 1(**0-3**)
- Group 2(**4-12**)
- Group 3(**13-19**)
- Group 4(**invalid**)

Boundary Value Analysis

Boundary Value Analysis

- Test cases are designed based on boundary values
- Conceptually, BVA is about testing the edges of equivalence classes
- Both valid boundaries (in the valid partitions) and invalid boundaries (in the invalid partitions)
- Most mistakes are made at the edges

Boundary Value Analysis(2)

Example:

A field for age is tested

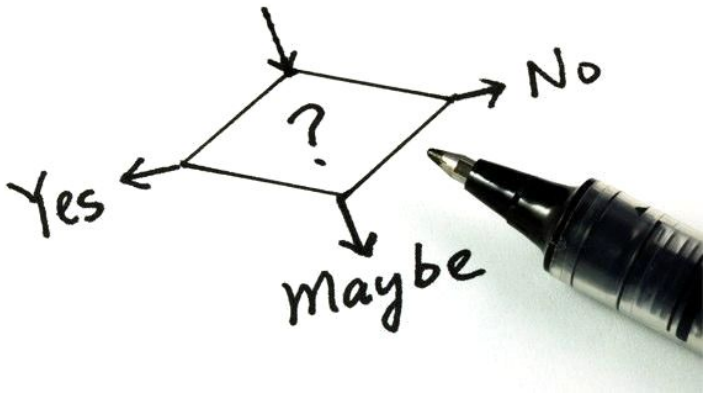
- if age is between **0-3 - baby**
- if age is between **4-12 - kid**
- if age is between **13-19 - teenager**
- etc...

Tests:
Group 1: -1,0,3,4
Group 2: 3,4,12,13
Group 3: 12,13,19,20

Decision Table Testing

Decision Table Testing

- **Method for testing business logic**
- Good way to deal with combinations of things (e.g. inputs)
- Also referred to as a ’cause-effect’ table
- Provide a systematic way of stating complex business rules, which is useful for developers as well as for testers.



Decision Table Testing (2)

Conditions	1	2	3	4	5	6	7	8
Condition A	T	T	T	T	F	F	F	F
Condition B	T	T	F	F	T	T	F	F
Condition C	T	F	T	F	T	F	T	F
Actions								
....								

Under this **concrete** combination of conditions - carry out this **concrete** combination of actions.

Decision Table Testing (3)

$2^n = 8$

$n = 3$

Conditions	1	2	3	4	5	6	7	8
Condition A	T	T	T	T	F	F	F	F
Condition B	T	T	F	F	T	T	F	F
Condition C	T	F	T	F	T	F	T	F
Actions								
....								

Decision Table Testing (3)

Example:

Conditions	1	2	3	4	5	6	7	8
New Customer (15%)	T	T	T	T	F	F	F	F
Loyalty card (10%)	T	T	F	F	T	T	F	F
Coupon (20%)	T	F	T	F	T	F	T	F
Actions								
Discount(%)	20	15	20	15	30	10	20	0

State Transition Testing

State Transition Testing

- Test cases are designed to execute valid and invalid state transitions
- When we have **sequences** of events that occur and conditions that apply to those events
- When the proper handling of a particular event/condition situation depends on the events and conditions that have occurred **in the past**

State Transition Testing (2)

How do we distinguish a state, an event, and an action?

State

- Persists until something external happens, usually triggering a transition
- A state can persist for an indefinite period

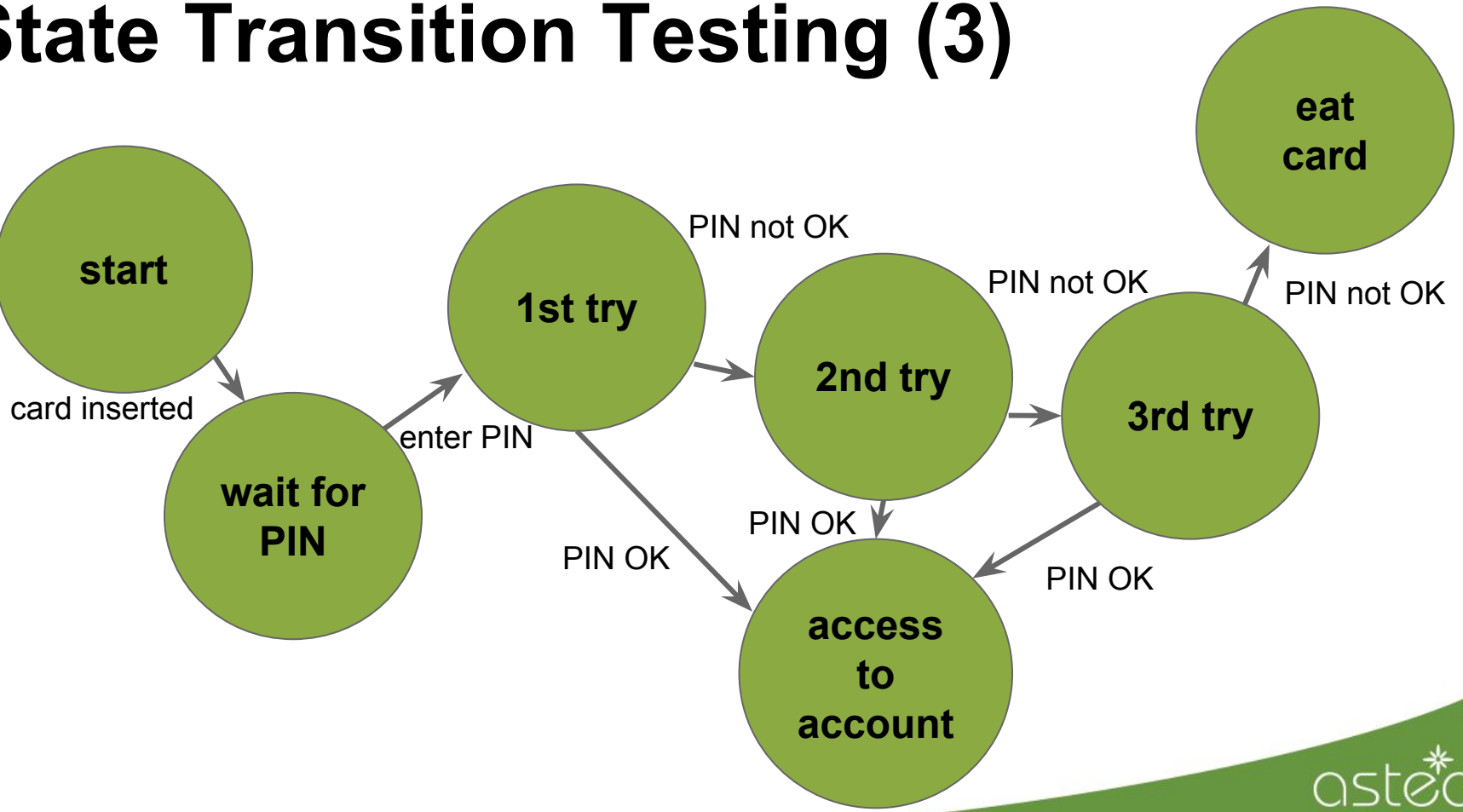
Action

- The response of the system during the transition
- An action, like an event, is either instantaneous or requires a limited, finite period

Event

- Occurs, either instantly or in a limited period
- It is something that happens

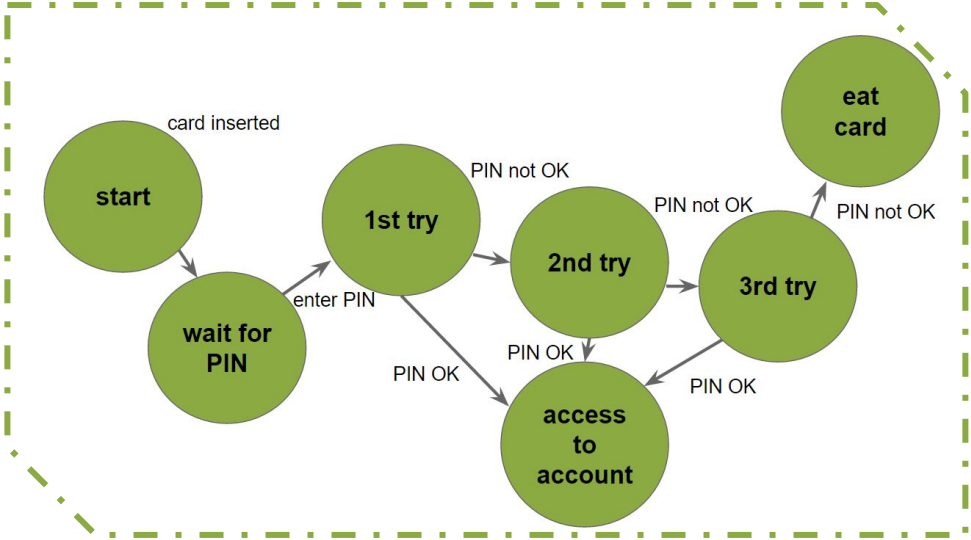
State Transition Testing (3)



State Transition Testing (4)

Test case examples:

- Correct PIN is entered the first time.
- Enter an incorrect PIN each time, so that the system eats the card.
- The PIN was incorrect the first time but OK the second time
- The PIN was correct on the third try.
- etc...



State Transition Testing (5)

State Transition table

Current State	Event	Action	Next State
start	card is inserted	Please enter PIN	wait for PIN
wait for PIN	enter PIN	Checking PIN	1st try
...	…	...	...

Use Case Testing

Use Case Testing

Use case testing is a technique that helps us identify test cases that exercise the whole system from start to finish.

- Description of a particular use of the system by an **actor** (a user of the system).
- It describes the interactions the actor has with the system in order to achieve a specific task
- Actors are generally people but they may also be other systems.
- Use cases are a **sequence of steps** that describe the interactions between the actor and the system.
- They serve as the foundation for developing test cases mostly at the system and acceptance testing levels.

Pairwise Testing

Pairwise Testing

- Testing the software using combinatorial method
- It's a method to test all the possible combinations of the parameters involved
- usually used in Configuration testing
- Most of the bugs are found with one or two options involved
  - The basic bug hypothesis is that this level of coverage is sufficient
- Tools for automatic generation of all-pairs are available

Example:

Check-Box - Checked or Unchecked
Radio Button - ON or OFF

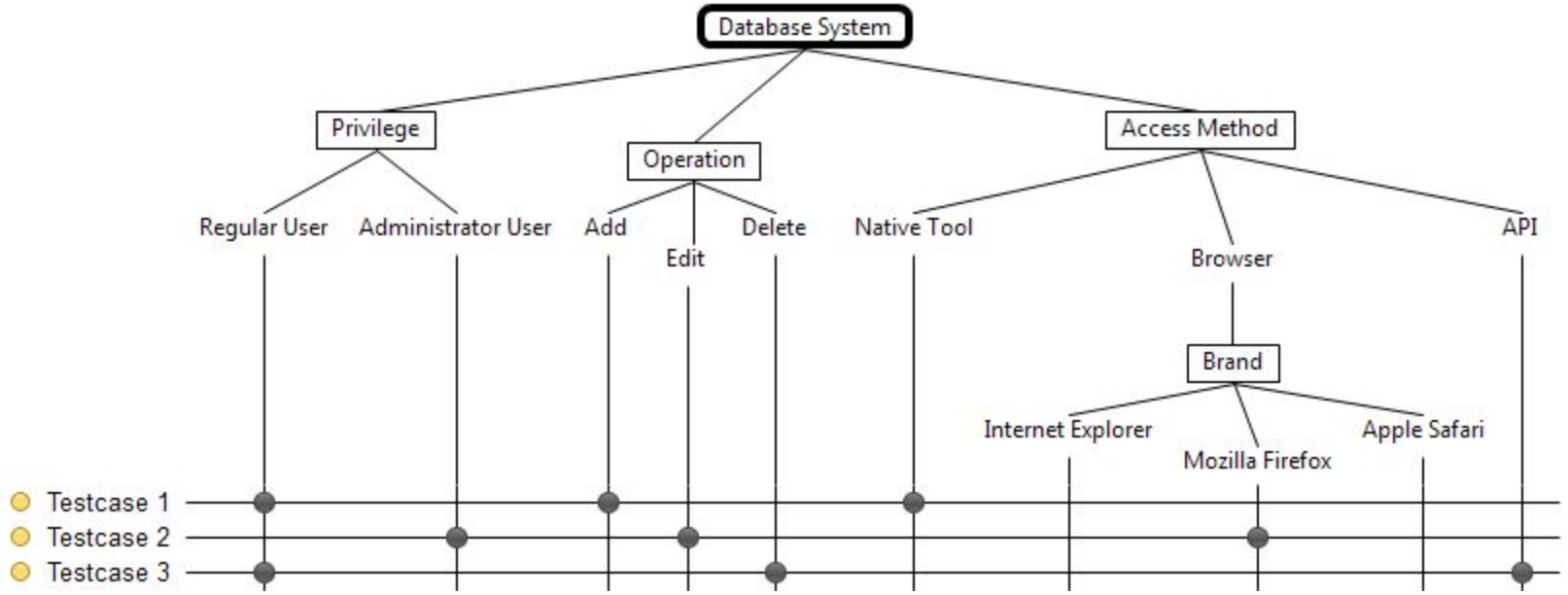
Classification Trees Testing

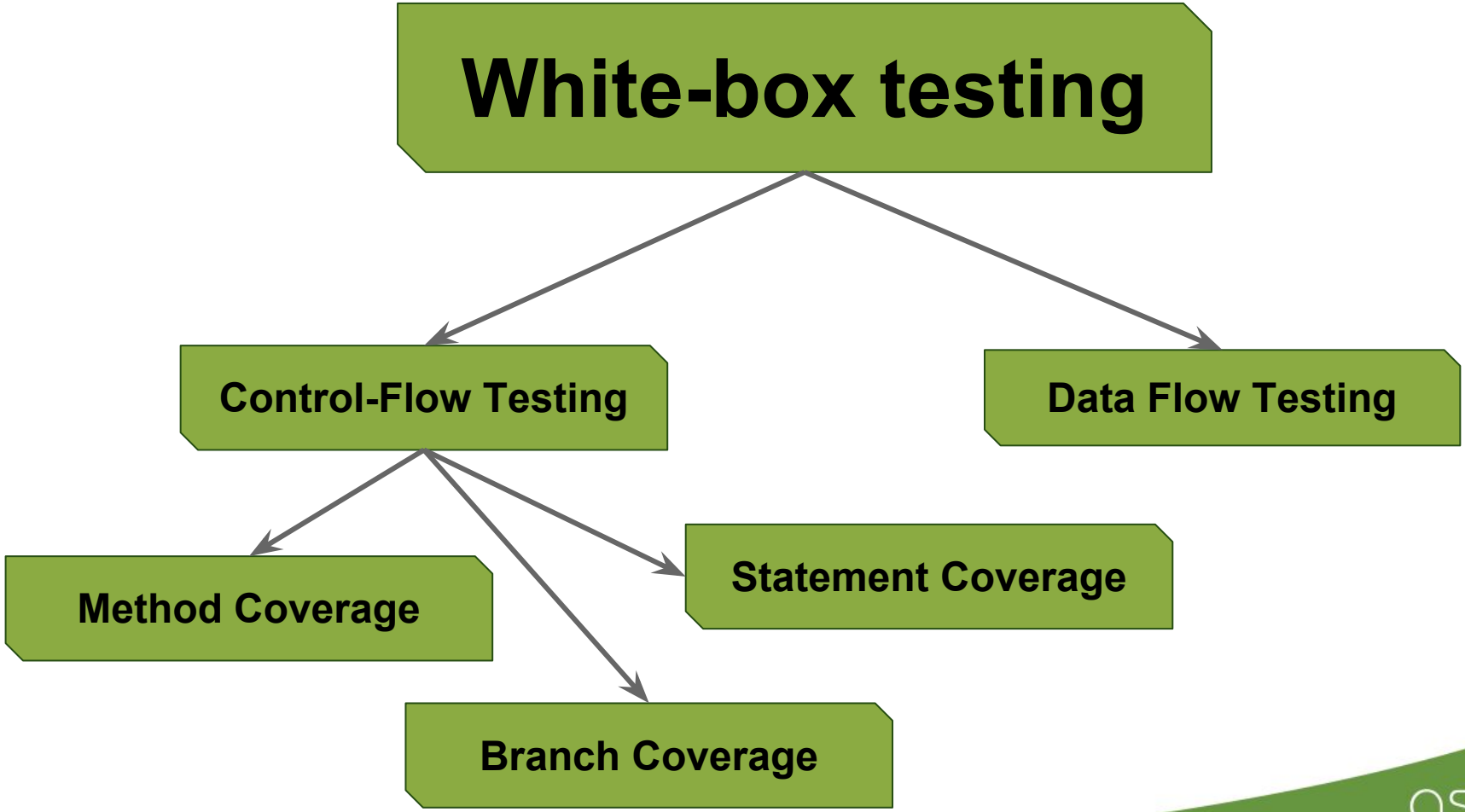
Classification Trees Testing

A tree showing equivalence partitions hierarchically ordered

- Test cases are designed to execute combinations of inputs and/or outputs
- Based on the functional specification of the test object
- Used to design test cases in the classification tree method
- Some options for one factor won't coexist with options for another factor
- Usually a tool is used (CTE)

Classification Trees Testing (2)





White-box Testing

White-box testing

- White-box techniques are a way to derive test cases based on analysis of the code
- Design test cases that:
  - Exercise independent paths within a module or unit
  - Exercise logical decision
  - Execute loops at their boundaries
- Also called clear box, structural or glass box testing
- It uses the internal structure of the component or system
- The code of the tested object is considered
- Testing can be commenced at an earlier stage. It’s applicable to:
  - Unit testing
  - Integration testing
  - System testing

Control-Flow Testing

Control-flow testing

- Represented in a flow graph
- We consider various aspects of this flowgraph in order to ensure that we have an adequate set of test cases
- The adequacy of the test cases is often measured with a metric called **coverage**
- Coverage is a measure of the completeness of the set of test cases

Control-flow testing (2)

Method Coverage

A measure of the percentage of methods that have been executed by test cases. Undoubtedly, your tests should call 100% of your methods

Statement Coverage

A measure of the percentage of statements that have been executed by test cases. Your objective should be to achieve 100% statement coverage through your testing.

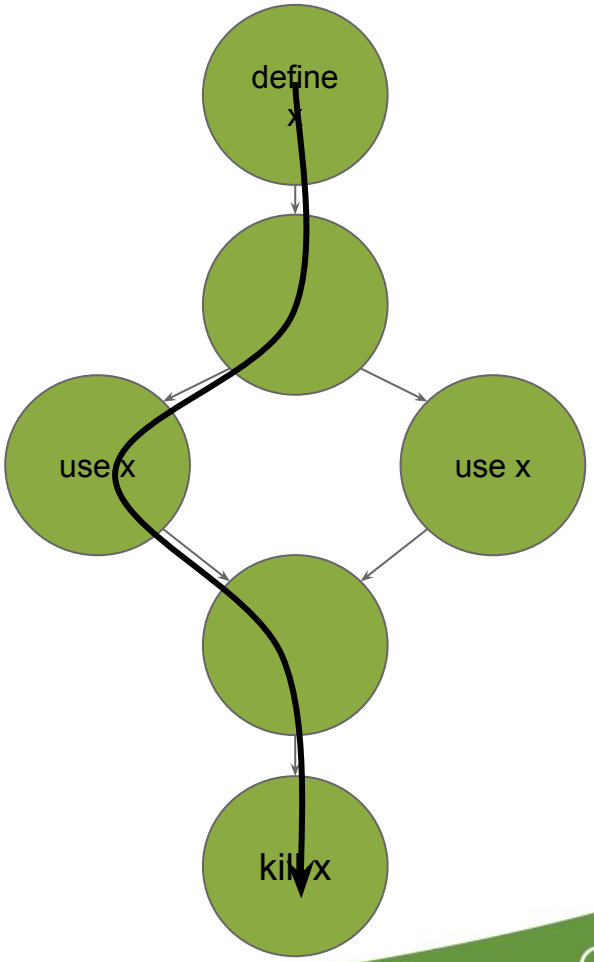
Branch coverage

A measure of the percentage of the decision points (Boolean expressions) of the program have been evaluated as both true and false in test cases.

Data-Flow Testing

Data-flow testing

- In data flow-based testing, the control flowgraph is annotated with information about **how** the program **variables are defined and used**
- Different criteria exercise with varying degrees of precision how a value assigned to a variable is used along different control flow paths.



Data-flow testing

Examples:

~d the variable does not exist (indicated by the ~), then it is defined (d)
~u the variable does not exist, then it is used (u)
~k the variable does not exist, then it is killed or destroyed (k)
dd Defined and defined again—not invalid but suspicious. Probably a programming error.
du Defined and used—perfectly correct. The normal case.
dk Defined and then killed—not invalid but probably a programming error.
uu Used and used again—acceptable.
uk Used and killed—acceptable.
kd Killed and defined—acceptable. A variable is killed and then redefined.
ku Killed and used—a serious defect. Using a variable that does not exist or is undefined is always an error.
kk Killed and killed—probably a programming error.

Summary

- What is Equivalence Partitioning?
- When do we use BVA?
- What is Use Case?
- How to make a decision table?
- What are the properties of State transition testing?
- What is Control-flow testing?
- Give examples for data-flow?

QUESTIONS