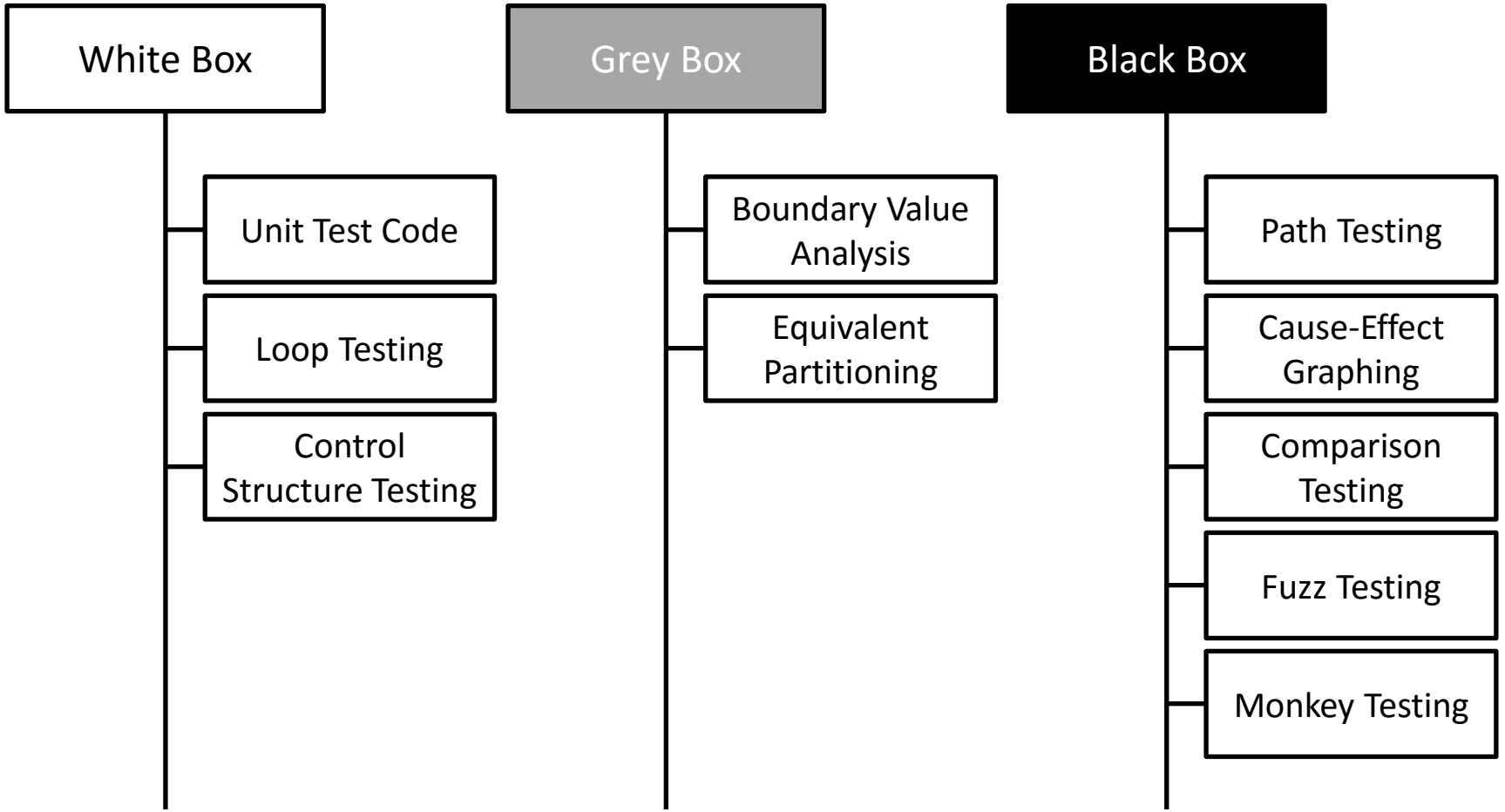
Software Testing

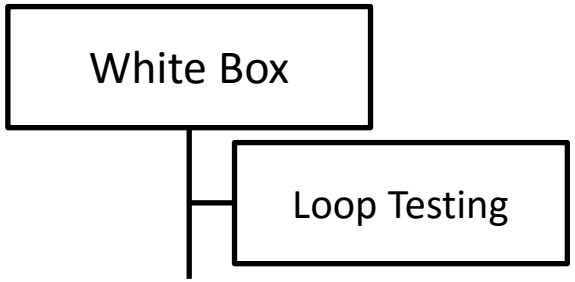
Test Techniques

by J. Janvier (Jay/文宇)

Test Techniques



Loop Testing



1. Identify ‘loops’ in the code,
2. Start testing the inner loop and then outward.

Loop Testing

Loop Testing on Code:

```
public class loopdemo
{
    // Compute total of numItems positive numbers in the array.
    // @param numItems how many items to total, maximum of 10.

    private int[] numbers = {5,-3,8,-12,4,1,-20,6,2,10};

    public int findTotal(int numItems)
    {
        int total = 0;
        if (numItems <= 10)
        {
            for (int count=0; count < numItems; count = count + 1)
            {
                if (numbers[count] > 0)
                {
                    total = total + numbers[count];
                }
            }
         }
         return total;
    }
}
```

Loop Testing

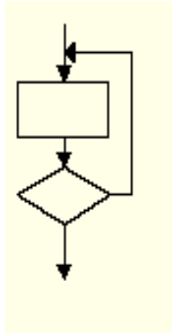
Loop Testing on Code:

```
public void testOne()
{
    loopdemo app = new loopdemo();
    assertEquals(0, app.findTotal(0));
    assertEquals(5, app.findTotal(1));
    assertEquals(5, app.findTotal(2));
    assertEquals(17, app.findTotal(5));
    assertEquals(26, app.findTotal(9));
    assertEquals(36, app.findTotal(10));
    assertEquals(0, app.findTotal(11));
}
```

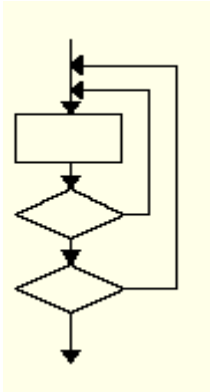
1st Loop : You test whether all variables initialize correctly,
2nd Loop : You test the functionality of the loop,
From 2nd Loop : You test the robustness of the loop and the limits.

Note: Loop Testing on Code is only useful if you can control the amount of looping.

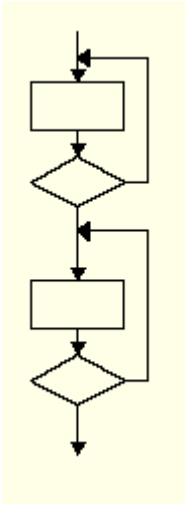
Loop Testing



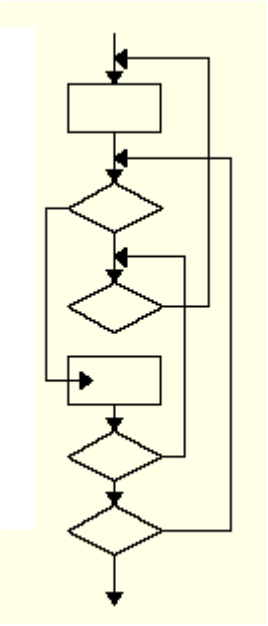
- Simple Loop;
  Single loop in software



- Nested Loop;
  Loop within a loop

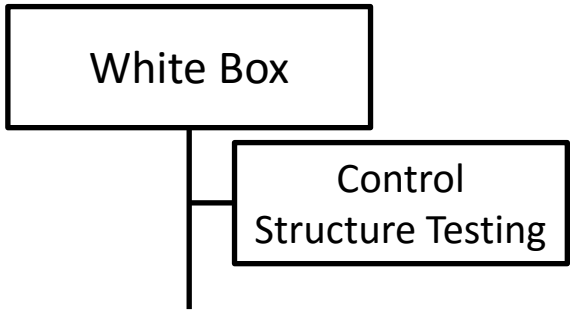


- Concatenated Loop;
  Sequence of loops



⇦ • Unstructured Loop;
  Don't Test; Request redesign

Control Structure Testing



Control Structure Testing are a set of smaller tests which aim to test control structures in code.

Control Structure Test Technique: Condition Testing

1. Identify decision making code (If statements, switch-case statements),
2. Test all the possible decisions/outcomes.

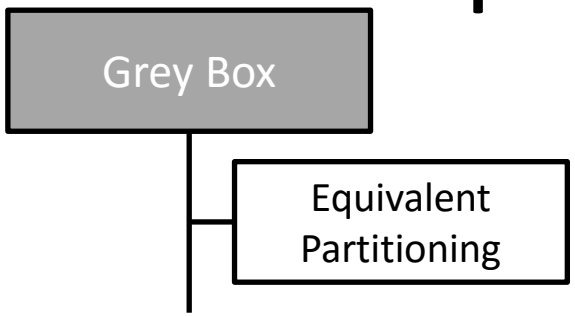
Control Structure Testing

Condition Testing

```
BEGIN
    ...
    IF sales > 50000 THEN
        bonus := 1500;
    ELSIF sales > 35000 THEN
        bonus := 500;
    ELSE
        bonus := 100;
    END IF;
    INSERT INTO payroll VALUES (emp_id, bonus, ...);
END;
```

Tests would include testing all possible outcomes of the decision

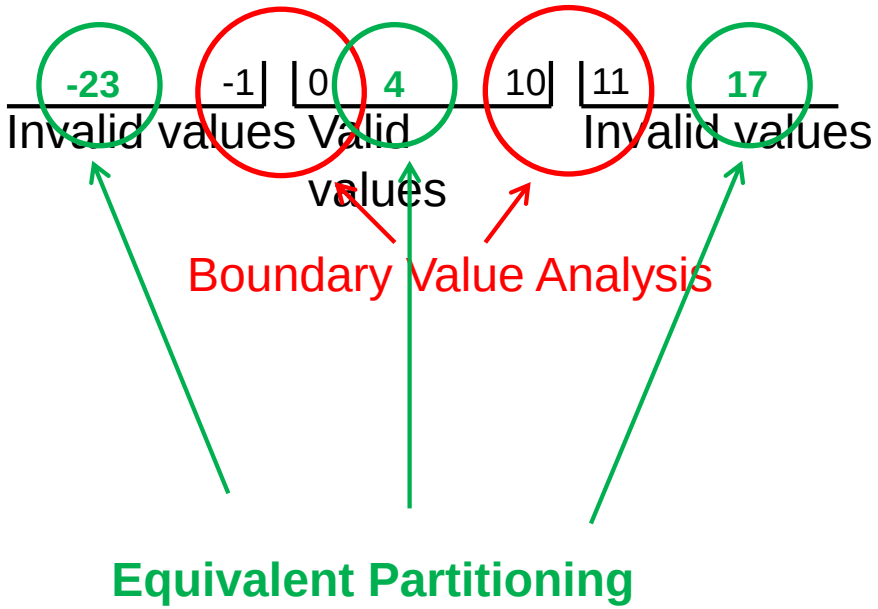
Equivalent Partitioning



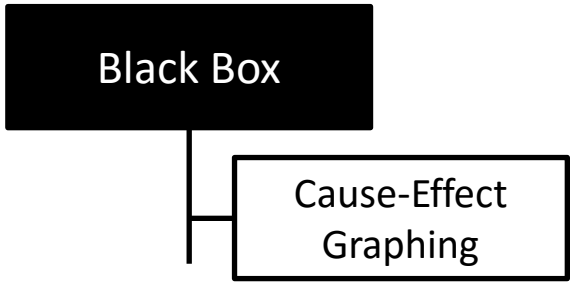
Test random values in each valid/invalid range of values (partition).

1. Define the valid/invalid ranges,
2. Test one random value from each range

(Often used in combination with Boundary Value Analysis)



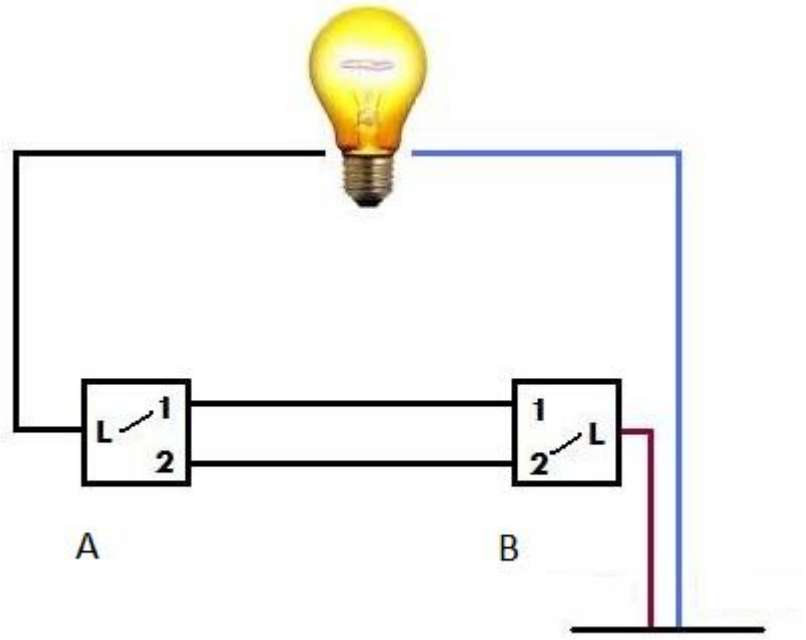
Test Techniques



1. Define combinations of input settings (matrix/diagram),
2. Test all combinations

Cause-Effect Graphing

Example 1

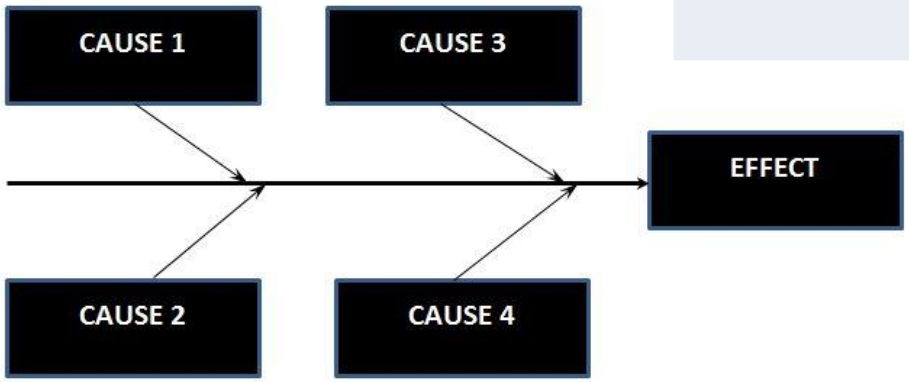


Cause (Switch A) +	Cause (Switch B)	Effect (Light)
L-1	L-1	On
L-2	L-1	Off
L-1	L-2	Off
L-2	L-2	On

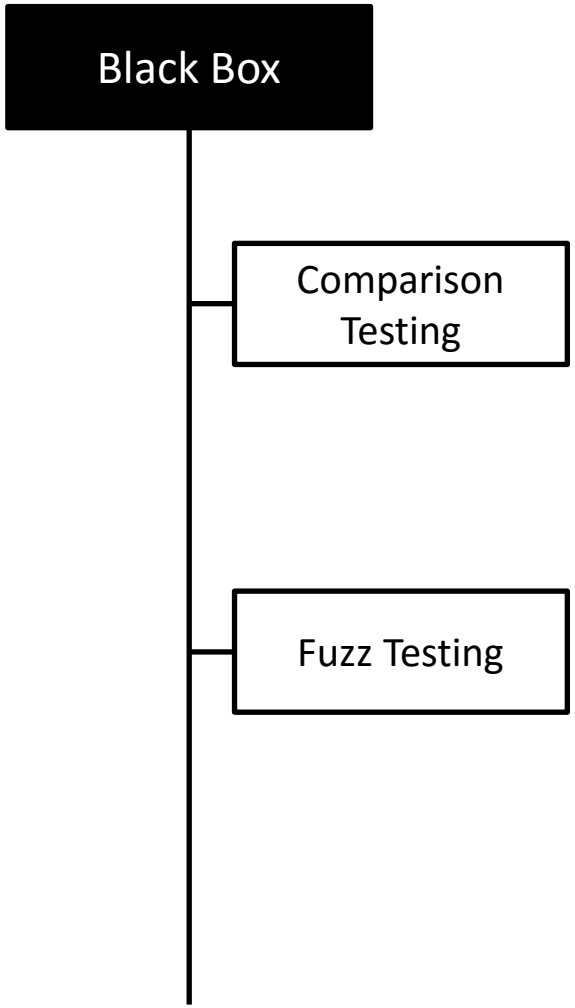
Cause-Effect Graphing

Example 2

Save Button	Filename	Space on Disk	Effect
Clicked	Defined	Not enough	Warning
Clicked	Undefined	Not enough	Warning
Clicked	Defined	Enough	Saved
Clicked	Undefined	Enough	Dialog
Not Clicked	Defined	Not enough	No Effect
Not Clicked	Undefined	Not enough	No Effect
Not Clicked	Defined	Enough	No Effect
Not Clicked	Undefined	Enough	No Effect



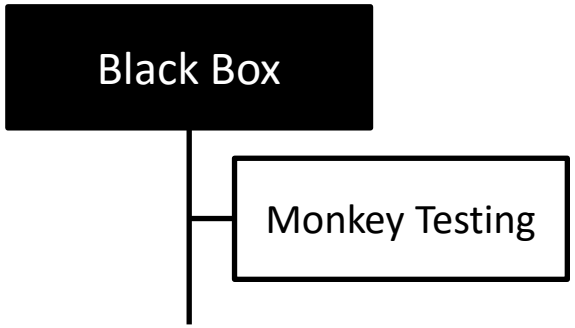
Test Techniques



1. Two teams create their own versions of the software,
2. Both versions are tested with the same test scenarios,
3. Test results are then compared, to comfirm the output

1. Feed random input values to the application,
2. If application crashes or hangs, the test failed (Used to test for Buffer Overflow errors, cross-site scripting, DOS attacks, formatting, etc.)

Monkey Testing



Generating events without regard to actual usage scenarios (example: randomly clicking arround)

Questions?