LECTURE 6 – TESTING (PART 2)

Master of Applied Computing

COMP-8117 : Advanced Software Engineering Topics

Dr. Aznam Yacoub – aznam.yacoub@uwindsor.ca
School of Computer Science

SCHEDULE

- Introduction
- Static Tests
- Dynamic Tests
- Conclusion

University of Windsor

TESTS CLASSES

- Two categories of tests:
  - Static tests => these tests work on the code without executing it (we talk about proofs).
  - Dynamic tests => these tests execute the software on a particular subset of inputs.

University of Windsor

TESTS CLASSES

- Two categories of tests:
  - Static tests => these tests work on the code without executing it (we talk about proofs).
  - Dynamic tests => these tests execute the software on a particular subset of inputs.
    - Verification and Validation processes

University of Windsor

MULTI-LEVEL TESTING

- Quality of dynamic testing depends on the quality of test cases. Do the test cases represent the use cases of the system ?
- Three kind of level:
  - Functional testing or Black-box
  - Grey-box
  - Structural testing or White-box

University of Windsor

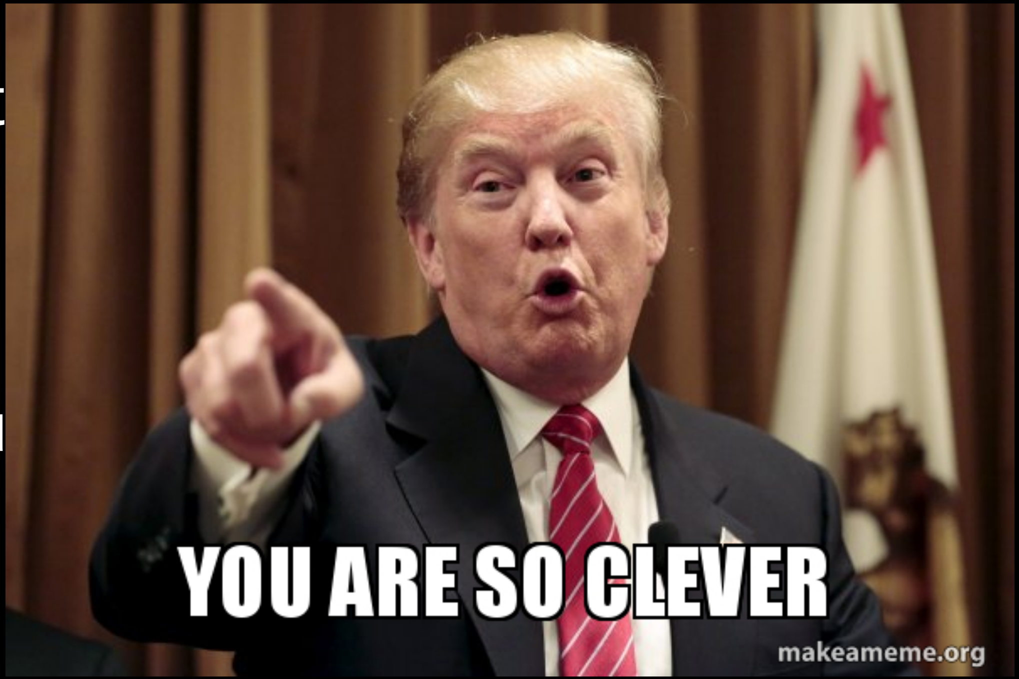
WHY THREE STRATEGIES ?

- Dynamic tests are based on execution of the software. It’s possible that some parts of code are never executed.
- In functional tests, we assume that we don’t know the code. These tests are defined before designing the program.
- In structural tests, we assume that we know the code. These tests are defined after designing the program.

University of Windsor

WHY THREE STRATEGIES ?

- If you apply functional tests only, you won't discover errors in non-executed parts => You need structural tests.
- If you apply structural tests only, you won't discover errors in unspecified parts => You need functional tests.



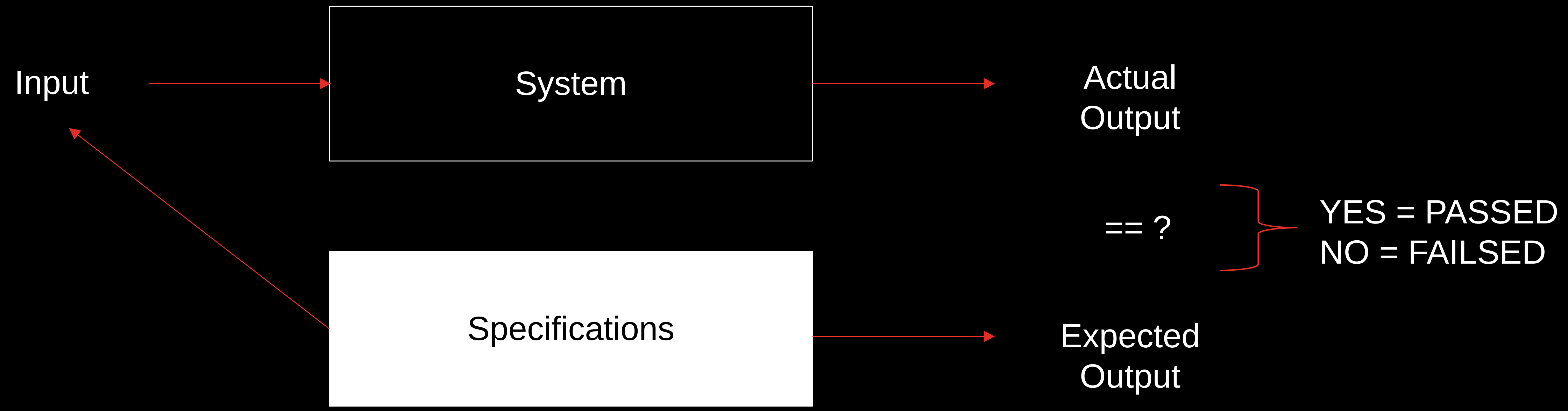
BLACK-BOX TESTING

- Data-driven, input/output-driven testing
- See the program as a black-box
  - You don’t see the code, you know only the requirements and specifications
  - Focus on finding scenarios/conditions in which the program does not behave according to its specifications while you don’t care about the internal behaviour and internal structure.
- Answer one question:
  - Does the system behave as predicted by its specifications ?

Black Box Testing Methods

University of Windsor

BLACK-BOX TESTING



Compare the actual output to the expected output.
No specification = All the behaviours are correct.

BLACK-BOX TESTING

- Write at least one test case per functional capability
  - User input validation
  - Output results
  - State transitions
  - Boundary and limit cases
- Iterate on functionalities until all tests pass
- Automate this process as much as possible

Black Box Testing Methods

University of Windsor

BLACK-BOX TESTING

- Example : For a function *find_minimum(A, B, C : real) : real*

1. Data : Parameters A, B, C (infinite possible values)
2. Possible types of significative cases:
    1. A < B <= C, A > B <= C… (possible permutations)
    2. A < 0… (Positive or negative)

University of Windsor

BLACK-BOX TESTING

- Example : For a function *find_minimum(A, B, C : real) : real*

3. For each type, try to find a good partition which covers all the cases

    1. A < B
    2. A < C
    3. B < C
    4. A positive
    5. A negative

    ….

University of Windsor

BLACK-BOX TESTING

- Example : For a function *find_minimum(A, B, C : real) : real*

4. Make a cartesian product of each tuple of partitions => Equivalent test classes

    1. A < B < C and A Positive
    2. A < B < C and A Negative

    ….

University of Windsor

BLACK-BOX TESTING

- Example : For a function *find_minimum(A, B, C : real) : real*

5. For each equivalent class, define test input
   1. A = 2, B = 3, C = 4
   2. A = -2, B = 3, C = 4
   3. Limit cases : A = B < C, B = C < A, A = C < B, A = B = C

University of Windsor

BLACK-BOX TESTING

- Example : For a module *stack* with two operations push and pop
- Alternate push and pop in different combinations and check the state of the stack using getters
  - Limit cases : what happens when you try to pop an empty stack
  - Limit cases : what happens when you try to push on a full stack

University of Windsor

BLACK-BOX TESTING

- In summary:
  - Test data derived solely from specifications
  - No knowledge of the internal structure
  - Exhaustive testing = use of every possible input conditions
    - Possible on a program, but not on a software
    - But you don't know the program
  - Result: Exhaustive testing is impossible.

WHITE-BOX TESTING

- Logic-driven
- See the program as a white-box
  - You know the code, you can examine the internal structure of the program and the architecture
  - Derive test data from examination of the logic of the program (neglect the specifications)
- Answer one question:
  - How the system behaves ?

White Box Testing Methods

WHITE-BOX TESTING

- You have a complete knowledge of the program:
  - Format of test is always the same : input, expected output, actual output
- Look at:
  - Code coverage : Must make every statement in the program execute at least once => Exhaustive path finding
  - Proper error handling (Exception management, return code…)
  - Working as documented (thread-safe, …)
  - Proper handling of resources (pointers…)
    - What happens when resources become constrained ?
- White-box testing may be exhaustive !

WHITE-BOX TESTING

- Problems:
  - The number of unique logic paths through a program can be huge
  - Even if you do an exhaustive path test, are you sure that you program match its specifications ? (What happens if something is not specified ?)
  - What about missing paths ?
  - What about data sensitivity errors ? (for example on floatting values)

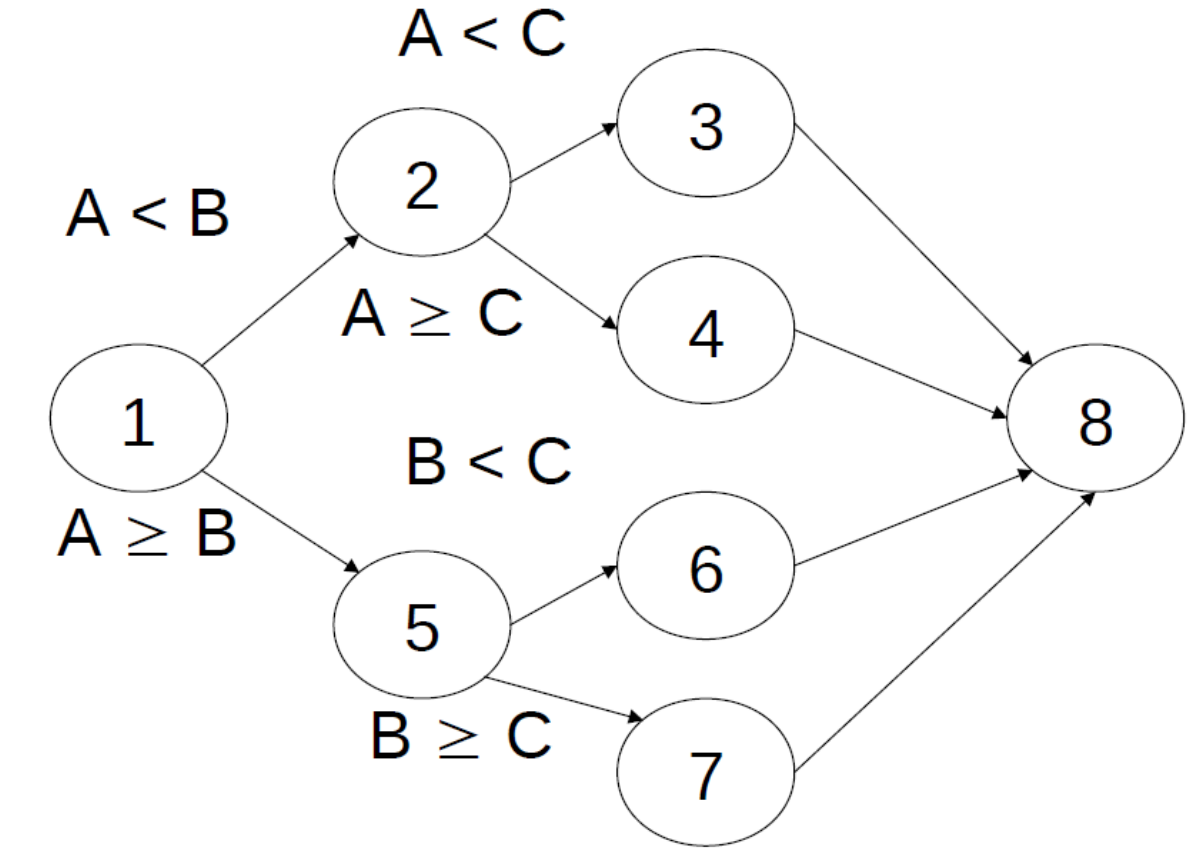
University of Windsor

WHITE-BOX TESTING

- Example: find_minimum(A, B, C)

```
If A < B
    if A < C
        return A
    else
        return C
Else
        …
```


WHITE-BOX TESTING



WHITE-BOX TESTING

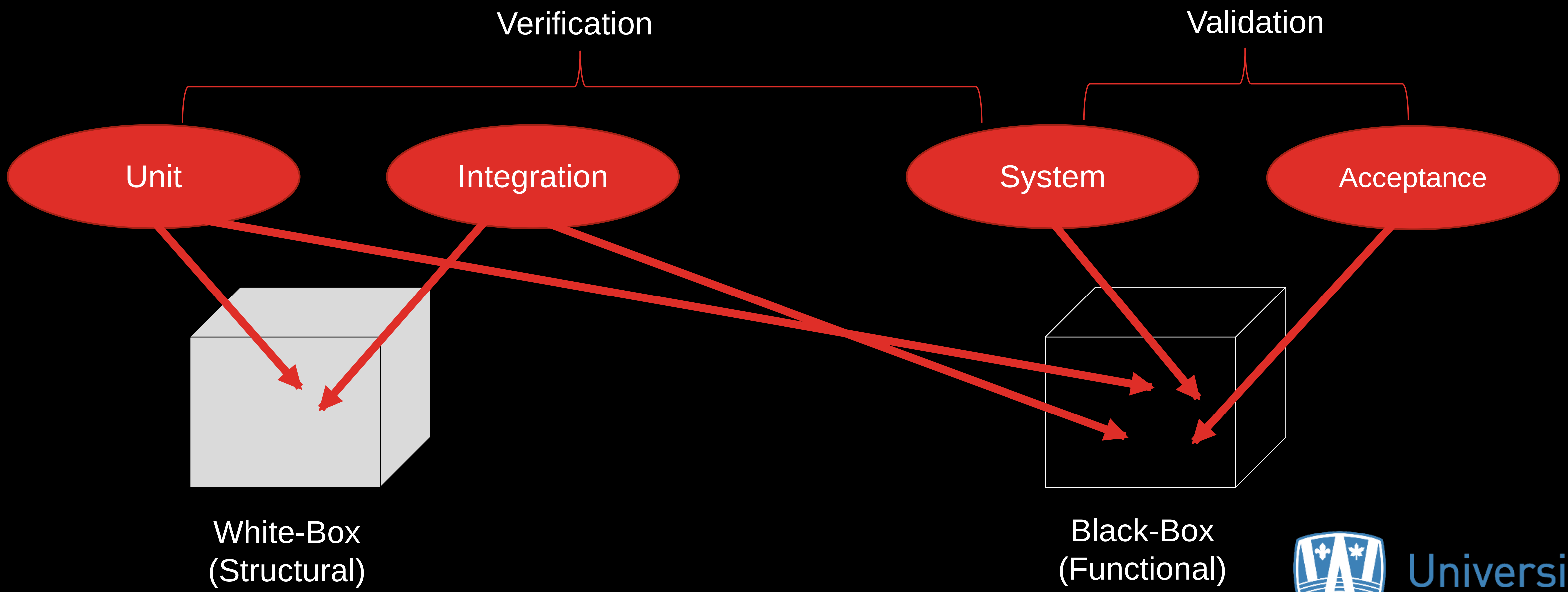
- Choose values for A, B, and C such that you’re sure to go through all the possible edges, paths, …
- You can use covering analysis tools to ensure that you test cases cover all your code.

University of Windsor

GREY-BOX TESTING

- Intermediate level between Black and White
  - Use knowledge of architecture to create a more complete set of black box tests
- Verify auditing and logging information (printf)
- Check data destined for other systems
- System-added informations (timestamps, checksum, etc)
- Looking for scraps : temporary files cleanup, memory leaks, data duplication/deletion

DYNAMIC TESTS SUMMARY



GREY-BOX TESTING

- Intermediate level between Black and White
  - Use knowledge of architecture to create a more complete set of black box tests
- Verify auditing and logging information (printf)
- Check data destined for other systems
- System-added informations (timestamps, checksum, etc)
- Looking for scraps : temporary files cleanup, memory leaks, data duplication/deletion

University of Windsor

DESIGN AND IMPLEMENT DYNAMIC TESTS

- Black-Box : written during the specification and design phases
  - Analysis of specifications
  - Help for designers
  - Help to modify specifications for future reuse
  - Validation tools for customer
- White-Box : written during design and programming phases

University of Windsor

DESIGN AND IMPLEMENT DYNAMIC TESTS

- Replace or complete manual testing with automatic testing
  - Create programs which execute tests : tests can be directly inserted in the program or read from a file
  - Create testing and profiling tools in the tested program to check pre-/post-conditions
- However, cost:
  - Need to create a specific architecture for embedded testing
  - Embedded tests can interfere with the program

University of Windsor

OTHER DYNAMIC TESTS

- Randomized tests: test cases and scenarios are generated using a probabilistic law
- Make the test data uniform
- Complete with limit cases
- Allows decision of stopping test if reliability is enough

University of Windsor

OTHER DYNAMIC TESTS

- Regression testing: check if a new functionality or a maintenance creates new errors
- We check that existing behaviours are not broken
- Often consist of a set of actual observed production inputs and their archived outputs from past versions

OTHER DYNAMIC TESTS

- Failure tests : if a failure is observed, a test reproducing this failure is created to make sure that particular failure doesn't happen anymire
- Set of actual observed inputs that caused the failure and the archived outputs after the system is fixed

University of Windsor

REFERENCES

This lecture is based on:

- Introduction to Software Testing – Ziad Kobti – University of Windsor
- Test – Amine Hamri – Aix-Marseille University
- CSCI3060U – Jeremy Bradbury - Ontario Tech University
- CSCI 5828 – Kenneth Anderson
- The Art of Software Testing – Glenford Myers

University of Windsor