Software Systems Verification and Validation

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University

2024-2025

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

(Next)/Today Lecture

- Test case design -White-box testing

- Continuous integration - Jenkins

Outline

- Testing - fundamental questions
- Testing strategy
- Levels of testing – Unit testing
- White -box testing
  - Control Flow Graph (CFG)
  - Cyclomatic complexity
  - Logic-Coverage Testing [Mye04] (statement, branch/decision, condition, decision-condition , multiple-condition coverage)
    - Path coverage criterion [NT05] (All-Path, Statement, Branch , Predicate Coverage Criterion)
    - Additional White box test design approaches [CB03] (Independent Path , Loop testing)
  - Advantages/Disadvantages
- Example - White-box testing
  1. Design of the test cases
  2. Maven project
  3. Jenkins – continuous integration tool

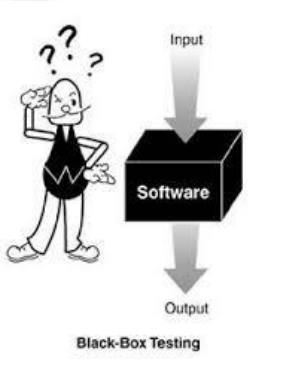
Testing - fundamental questions

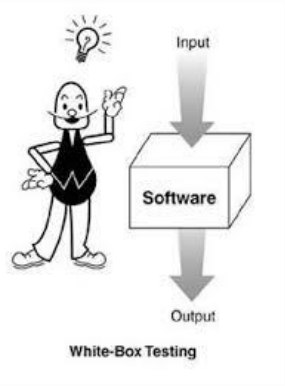
- **Why do we test?**
  - We test a product to learn about its quality. [BBST]
- **How do we organize the process of testing?**
  - Testing strategy problem
- **When have we tested enough?**
  - Testing measuring problem

Testing strategies [BBST]

- **Testing strategy is:**
  - The guiding framework for deciding what tests (what test techniques) are best suited to your product.
  - **Context** and **information objectives** are (or should be) the drivers of any **testing strategy.**

- **Selecting the Testing techniques ?**
  - Techniques differ in core.





Story ...

- Rewrite story ...
  Little Red Riding Hood!

Little Red Riding Hood!

- Story
  - http://www.eastoftheweb.com/short-stories/UBooks/LittRed.shtml

- **Input: RRH, W**
  - **Preconditions: RRH shouldn't tell strangers her direction.**
- **Result: r**
  - **Postconditions: (r=True and RRH shouldn't be late and RRH should arrive at grandma's house successfully) or (r=False and RRH is late at grandma)**
- **Algorithm NewRedRidingHood(RRH, Wolf, r) is:**

@ r= False
@ Red Riding Hood(RRH) receives basket for the grandma.
@ RRH starts the journey in the wood.
@ RRH meets the Wolf (W)
@ IF (W asked RRH about her direction)
@ RRH answers: "To my grandmother's!"
@ W suggested to pick up flowers.
@ If (RRH decides to pick up flowers)
@ She is late for her grandma.
@ W eats her grandma.
@ r = False
@ Else
@ She is not late for her grandma.
@ W does not eat her grandma.
@ RRH arrives at grandma's house successfully.
@ r = True
@ Else
@ r = True

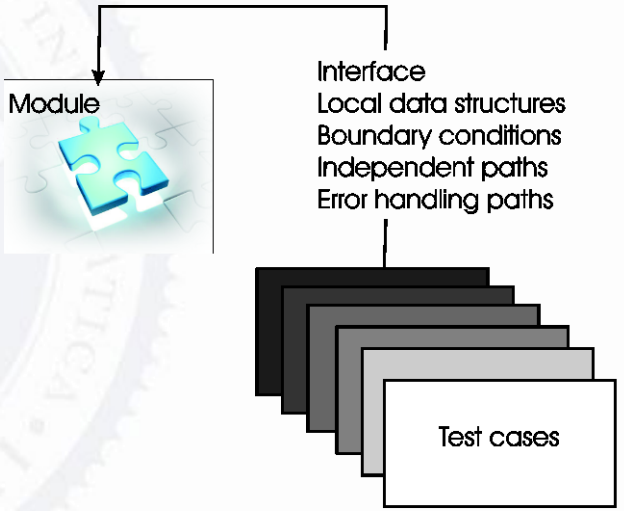
- **Is the correct algorithm for "safe" (r=True) version of the story?**

- Bug 1: RRH answers: "To my grandma" (actual result) but the expected result (RRH shouldn't tell strangers her direction)
- Bug 2: RRH decides to pick up flowers, so she is going to be late and thus the grandma is eaten

Levels of testing – Unit testing

Test case design

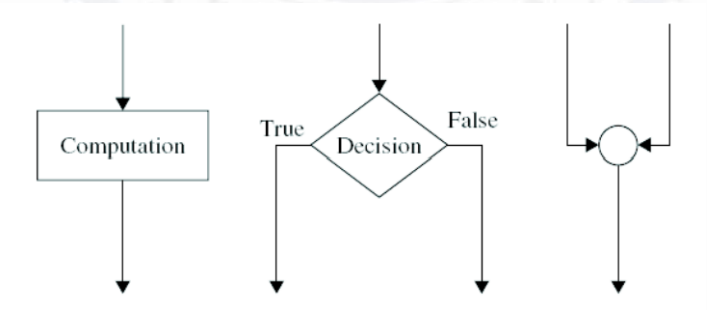
- Information needed when designing test cases for a module:
  - specification of the module
  - the module's source code
- Test case design procedure for a module test is:
  - Applying black-box methods to the module's specification.
  - Analyze the logic of the module using white-box methods.



White-box testing

Control Flow Graph

- A Control Flow Graph (CFG) is a graphical representation of a program unit.
- A CFG has exactly one entry node and exactly one exit node.
- Three symbols are used to construct a CFG
  - nodes - sequential statements, decision and looping predicates
  - edges - represent transfer of control
- Path in the CFG [NT05] - is represented as a sequence of computation and decision nodes from the entry node to the exit node.
- An independent path [CB03] is any path through the program that introduces at least one new set of processing statements or a new condition.



White-box testing

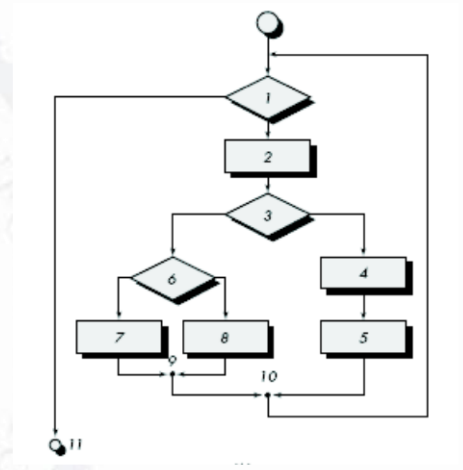
Cyclomatic complexity

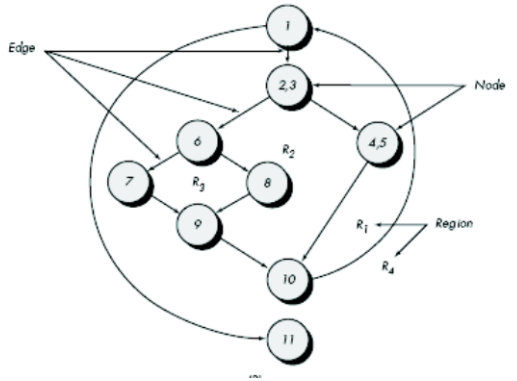
- Cyclomatic complexity
  - The number of independent paths in the basis set of a program and provides us with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.
  - CC = The number of regions of the flow graph.
  - CC = E - N + 2, where E - #edges, N - #nodes.
  - CC = P+1, where P - #predicate nodes

White-box testing

Cyclomatic complexity

- CC
  - CC = four regions = 4.
  - CC = 11 edges - 9 nodes + 2 = 4.
  - CC = 3 predicate nodes + 1 = 4.
- A set of independent paths:
  - path 1: 1-11.
  - path 2: 1-2-3-4-5-10-1-11.
  - path 3: 1-2-3-6-8-9-10-1-11.
  - path 4: 1-2-3-6-7-9-10-1-11.





Outline

- Testing - fundamental questions
- Testing strategy
- Levels of testing – Unit testing
- White -box testing
  - Control Flow Graph (CFG)
  - Cyclomatic complexity
  - Logic-Coverage Testing [Mye04] (statement, branch/decision, condition, decision-condition , multiple-condition coverage)
    - Path coverage criterion [NT05] (All-Path, Statement, Branch , Predicate Coverage Criterion)
    - Additional White box test design approaches [CB03] (Independent Path , Loop testing)
  - Advantages/Disadvantages
- Example - White-box testing
  1. Design of the test cases
  2. Maven project
  3. Jenkins – continuous integration tool

White-box testing

Logic-Coverage Testing [Mye04]

- [Mye04] – "… the ultimate white-box test is the execution of every path in the program, but complete path testing is not a realistic goal for a program with loops…."
- Select the minimum number of test cases such that we achieve:
  1. statement coverage
  2. branch/decision coverage
  3. condition coverage
  4. decision-condition coverage
  5. multiple-condition coverage

White-box testing

Logic-Coverage Testing [Mye04]

1. Statement coverage (sc)

- **Goal**: to execute every statement in the program at least once.
- Complete statement coverage is the weakest coverage criterion in program testing.
  - Any test suite that achieves less than statement coverage for new software is considered to be unacceptable.

White-box testing

Logic-Coverage Testing [Mye04]

2. Decision (branch) coverage (dc)

- A branch is an ongoing edge from a node.
  - All the rectangle nodes have at most one ongoing branch, except the exit node.
  - All the diamond nodes have two outgoing branches.
- Covering a branch means selecting a path that includes the branch.
- Complete branch coverage means selecting a number of paths such that every branch is at least one path.
  - selecting enough number of paths such that every condition evaluates to true at least once and to false at least once.

White-box testing

Logic-Coverage Testing [Mye04]

2. Decision (branch) coverage (dc) - issues

- Remark: dc ➔ sc
  - Why? Since every statement is on one subpath emanating from a branch statement or from the entry point of the program, every statement must be executed if every branch direction is executed.
- Exceptions:
  - Programs with no decisions.
  - Programs with multiple entry points. A given statement might be executed only if the program is entered at a particular entry point.
- A branch with multiple conditions - some decisions may remain uncovered.
  - if (a == 2 || b > 1) < statement > .
  - if the second condition it was written b < 1 by mistake, then the test case with a=2 wouldn't discover the error!

White-box testing

Logic-Coverage Testing [Mye04]

3. Condition coverage (cc)

- **Goal**: to write enough test cases to ensure that each condition in a decision takes on all possible outcomes at least once.
- cc ➔ dc (in general).
  - cc may cause (but does not always) every individual condition in a decision to be executed with both outcomes.
- Exceptions:
  - if (A&&B) < statement >
  - cc ➔ TC1 for A true, B false, and TC2 for A false and B true
  - But the statement is not executed (dc for True is not covered!
  - ➔there is a need for decision/condition coverage

White-box testing

Logic-Coverage Testing [Mye04]

4. Decision/condition coverage (dcc)

- **Goal**: requires sufficient test cases that:
  - each condition in a decision takes on all possible outcomes at least once;
  - each decision takes on all possible outcomes at least once;
  - each point of entry is invoked at least once.
- dc ➔ cc (in general)
- Exceptions:
  - When certain condition mask other conditions
  - Results of conditions in && and || expressions can mask or block the evaluation of other conditions (i.e. if an && condition is false then none of subsequent conditions in the expression need to be evaluated)
  - Thus, errors in logical expressions are not necessarily revealed by the condition-coverage and decision/condition coverage criteria

White-box testing

Hierarchy of strengths for sc, dec, cdc

- From weakest to strongest: sc, dc, cdc.
- The implication for this approach to test design is that the stronger the criterion, the more defects will be revealed by the tests.
- In most cases the stronger the coverage criterion, the larger the number of test cases that must be developed to ensure complete coverage.
- ➔the tester must decide (based on the type of code, reliability requirements, resources available) which criterion to select!

White-box testing

Logic-Coverage Testing [Mye04]

5. Multiple condition coverage (mcc)

- Goal: write sufficient test cases that:
  - all possible combinations of condition outcomes in each decision, and
  - all points of entry are invoked at lest once.
- mcc ➔ dcc (in general)
- Remark: A set of test cases satisfying the multiple-condition criterion also satisfies the decision coverage, condition coverage, and decision/condition coverage criteria.

White-box testing

Minimum test criterion

- For programs containing only one condition per decision:
  - Test cases to evoke all outcomes of each decision at least once, and
  - Test cases to invoke each point of entry at least once, to ensure that all statements are executed at least once.
- For programs containing decisions having multiple conditions:
  - Test cases to evoke all possible combinations of condition outcomes in each decision, and
  - all points of entry to the program, at least once.

White-box testing

Path coverage criterion [NT05]

- [NT05] – "A path is represented as a sequence of computation and decision nodes from the entry node to the exit node."

1. All-Path coverage criterion
2. Statement coverage criterion
3. Branch coverage criterion
4. Predicate Coverage Criterion

White-box testing

Path coverage criterion [NTO5]

1. All-Path coverage criterion

- The all-path selection criterion
  - is desirable but it is difficult to achieve in practice
  - is achievable but not practical

➔ reduced number of paths.

- Structural criteria are applied based on statements, edges and paths.

White-box testing

Path coverage criterion [NTO5]

2. Statement coverage criterion

- See [Mye04]

White-box testing

Path coverage criterion [NTO5]

3. Branch coverage criterion

- See [Mye04]

White-box testing

Path coverage criterion [NTO5]

3. Predicate coverage criterion

- There is a need to design test cases such that a path is executed under all possible conditions.

- If all possible combinations of truth values of the conditions affecting a selected path have been explored under some tests, then we say that *predicate coverage* has been achieved.
  - Lecture - In Class Work
  - 5 minutes – Read document "PredicateCoverage.pdf" posted on Teams – Lecture 3.
  - Post the solution for Figure 4.9a) for Predicate Coverage.
  - Post the solution for Figure 4.9b) for Predicate Coverage.

White-box testing

Additional White box testing design approaches [CB03]

- [CB03] – "A path is represented as a sequence of computation and decision nodes from the entry node to the exit node."
    1. Independent Path coverage criterion
    2. Loop testing

White-box testing

Additional White box testing design approaches [CB03]

1. Independent Path coverage [CB03]

- Path in the CFG [NT05] - is represented as a sequence of computation and decision nodes from the entry node to the exit node.
- An independent path [CB03] is any path through the program that introduces at least one new set of processing statements or a new condition.
  - ➔Construct the set of independent paths for a graph.
    - ➔This set is called: ..... [CB03]
- ➔Remark:
  - ➔coverage based on independent path testing ? complete path coverage

White-box testing

Additional White box testing design approaches [CB03]

2. Loop testing [CB03]

- Simple loops - n is the maximum number of allowable passes through the loop:
  - Skip the loop entirely.
  - Only one pass through the loop.
  - Two passes through the loop.
  - m passes through the loop where m < n.
  - n-1, n, n + 1 passes through the loop.
- Nested loops
  - Start at the innermost loop. Set all other loops to minimum values.
  - Conduct simple loop tests for the innermost loop while holding the outer loops at their minimum iteration parameter .
  - Work outward, conducting tests for the next loop, but keeping all other outer loops at minimum values.
  - Continue until all loops have been tested.

White-box testing

Advantages	Disadvantages
• Code coverage • Testing can be commenced at an earlier stage. • Find the fault.	• A skilled tester is needed to carry out this type of testing. • No ambiguities in spec. may be found. • After code is written.

White-box testing

Example

- **Problem statement**: Compute the number of participants with the maximum score (0 to 100 points possible) at a competition.

- Applied:
  - Construction of the CFG.
  - CC metric
  - Coverage: statements, conditions/decisions, paths, loops.
- See example files
- **Lecture03_Demo**

- Maven
  - goal - to allow a developer to comprehend the complete state of a development effort in the shortest period of time
  - https://maven.apache.org/what-is-maven.html
- Jenkins
  - Continuous integration tool

White-box testing

Example

- Maven
  - goal - to allow a developer to comprehend the complete state of a development effort in the shortest period of time
  - https://maven.apache.org/what-is-maven.html
  - The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application.
    - https://maven.apache.org/surefire/maven-surefire-plugin/index.html
- Jenkins
  - Continuous integration tool

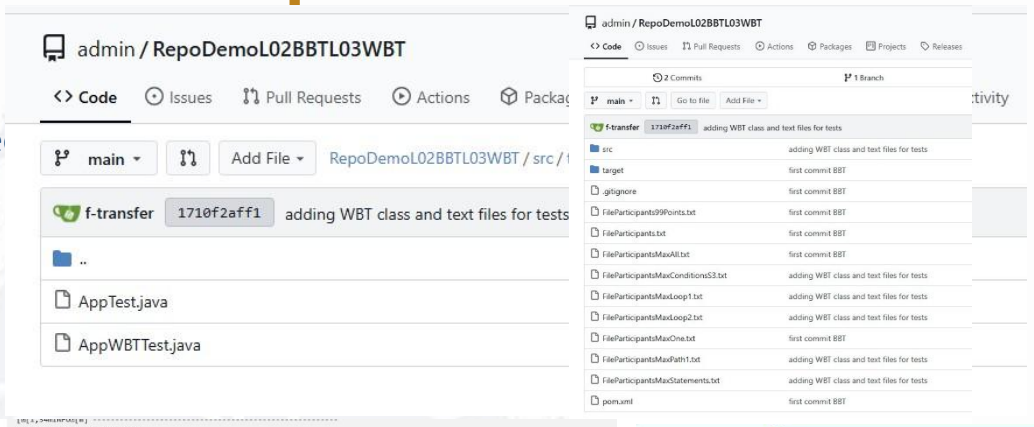
- File Info_CI_CD.pdf
  - Docker and Docker compose
- Create Repository in Gitea
- Add the source code to this project
- Create the Pipeline job
- Run the Job

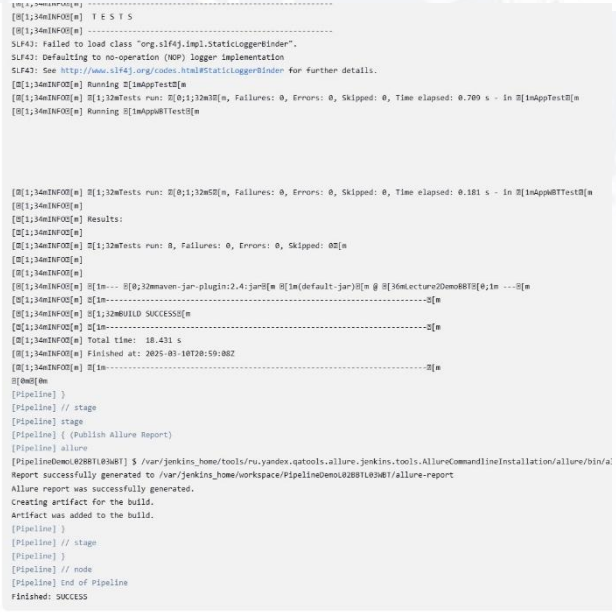
White-box testing

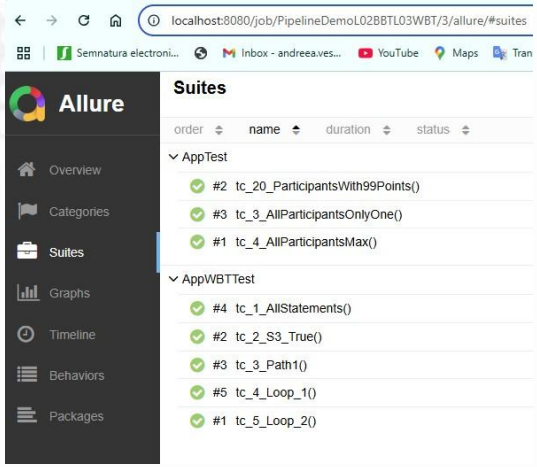
Example

- File Info_CI_CD.pdf
  - Docker and Docker compose
- Create Repository in Gitea
- Add the source code to this proje
- Create the Pipeline job
- Run the Job

```
d:\temp\ssvv2025\RepoDemoL02BBTL03WBT>git commit -m "adding WBT class and text files for tests"
[main 1710f2a] adding WBT class and text files for tests
 6 files changed, 86 insertions(+)
 create mode 100644 FileParticipantsMaxConditionsS3.txt
 create mode 100644 FileParticipantsMaxLoop1.txt
 create mode 100644 FileParticipantsMaxLoop2.txt
 create mode 100644 FileParticipantsMaxPath1.txt
 create mode 100644 FileParticipantsMaxStatements.txt
 create mode 100644 src/test/java/AppWBTTest.java
```







Outline

- Testing - fundamental questions
- Testing strategy
- Levels of testing – Unit testing
- White -box testing
  - Control Flow Graph (CFG)
  - Cyclomatic complexity
  - Logic-Coverage Testing [Mye04] (statement, branch/decision, condition, decision-condition , multiple-condition coverage)

    - Path coverage criterion [NT05] (All-Path, Statement, Branch , Predicate Coverage Criterion)

    - Additional White box test design approaches [CB03] (Independent Path , Loop testing)
  - Advantages/Disadvantages
- Surprise!
- Example - White-box testing
  1. Design of the test cases
  2. Maven project
  3. Jenkins – continuous integration tool

References

- [Pat05] R. Patton. Software Testing. Sams Publishing, 2005.
- [PY08] M. Pezzand and M. Young. *Software Testing and Analysis: Process, Principles and Techniques*. John Wiley and Sons, 2008.
- [Mye04] Glenford J. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., 2004
- [You79] E. Yourdon, Structured Walkthroughs, Prentice-Hall,Englewood Cliffs, NJ, 1979
- [NT05] K. Naik and P. Tripathy. *Software Testing and Quality Assurance*. Wiley Publishing, 2005.
- [CB03] Jean-Francois Collard and Ilene Burnstein. *Practical Software Testing*. Springer-Verlag New York, Inc., 2003.
- [Fre10] M. Frentiu, Verificarea si validarea sistemelor soft, Presa Universitara Clujeana, 2010
- [BBST] BBST Testing course, http://testingeducation.org/BBST/
  - **Foundations of Software Testing**
    - **Lecture 5: The Impossibility of Complete Testing**
- Tutorials - SSVV lecture's homepage.
  - www.cs.ubbcluj.ro/~avescan

Next Lecture (Today)

- Levels of testing

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)