# L21 - White-box testing

CS3028 - Principles of Software Engineering

**Ernesto Compatangelo**

Department of Computing Science

UNIVERSITY OF ABERDEEN



## 21.1 Reminding past issues and mapping them to current topics



### Where are we now?

⇒ · · · · · ·

⇒ Elaboration

⇒ Construction (third UP phase)

⇒ Software Quality Control

⇒ Software Testing

⇒ Black-Box Testing

⇒ White-Box Testing

⇒ · · · · · ·

## 21.2 Introducing white-box testing



### White-box testing: what is it?

- A technique that **derives test cases from program structure**, also called *glass-box testing* or *structural testing*
- W-B testing **focuses on control flow and on data flow**, checking whether they happen as expected:
  - Does a block of code execute when it should?
  - Does a variable receive the value it should?
- W-B testing uses a **graphical representation** (*flow graphs*) to lay out its strategy





### Motivations for white-box testing

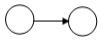
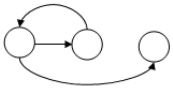
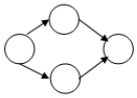
White-box testing uncovers a different class of errors compared to black-box:

- **Typographical errors** in the code
- **Errors in counterintuitive logical flows** (*e.g.*, nested and overcomplicated if-then-else structures)
- **Errors in poorly understood 'special cases'** (*e.g.*, exceptions)

## 21.3 Flow graphs



### Flow graphs

- Graphical representations of control flow in a program
- Flow graphs capture the three aspects of program *procedurality*, namely, **sequence**, **iteration** (for/while) , and **selection** (if/case)
- A **node** represents a sequence of statements
- An **edge** represent a possible flow of control
- An edge must terminate at a node (even if that node does not represent any statement)
- Sequence example: 
- Iteration example: 
- Selection example: 

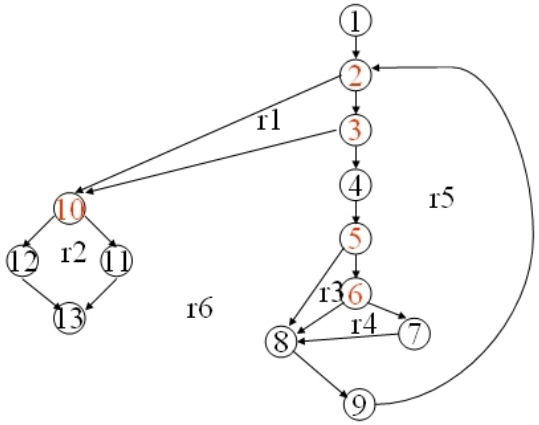




### Flow graphs: textual example

```
double average = 0, sum = 0;
int totalInput = 0, totalValid = 0;
myAvg(double[] value, double minimum, double maximum) {
     int i = 0;
     sum = 0;
     totalInput = totalValid = 0;                                    (1)
     while( (value[i]!= -999) && (totalIinput<100) ) {          (2) (3)
         totalInput++;                                               (4)
         if( (value[i]>=minimum) && (value[i]<=maximum) ) {     (5) (6)
             totalValid++;
             sum = sum + value[i];                                   (7)
         } // endif                                                  (8)
         i++;                                                        (9)
     } // endwhile                                                  (10)
     if (totalValid > 0) {                                          (11)
         average = sum/totalValid;
     } else {
         average = -999;                                            (12)
     } // endif                                                     (13)
}
```

## Flow graphs: diagrammatic example




1-2-10-11-13
1 - 2 - 10 - 12 - 13
1 - 2 - 3 - 10 - 11 - 13
1 - 2 - 3 - 4 - 5 - 8 - 9 - 2...
1 - 2 - 3 - 4 - 5 - 6 - 8 - 9 - 2...
1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 2...





## The notion of path

- A path is an instruction sequence that starts at the entrance of a routine (method) and ends at its exit
- An **independent path** is a path that
  - introduces at least one new instruction/condition
  - introduces at least one new edge in flow graph notations
- A **basis path set** is a set of independent paths that **ensures complete coverage**
- complete coverage DOES NOT mean all paths, but all the independent paths out of which all paths are created

## The cyclomatic complexity of a flow graph

- The Cyclomatic Complexity V(G) defines the **number of independent paths in a basis path set**
- V(G) also defines the **upper bound** on the number of **paths to be tested to ensure complete coverage**
- V(G) is computed using one of three different methods, *i.e.*
  - No of regions in a flow graph
  - No. of edges - No. of nodes + 2
  - No. of predicates + 1
- In our example:
  - No of regions = 6
  - No. of edges - No. of nodes + 2 = 17 - 13 + 2 = 6



### 21.4 White box testing method No 1: basis path



## White-box testing method example: basis path

- is a mandatory white-box technique
- aims at what is called **complete cover** by executing
  - every statement/instruction at least once
  - every branch from a condition at least once *(for code with 'goto' statements)*
  - is based on the notion of **path**

## Basis path test case design

- Construct a flow graph for the routine under testing
- Compute its cyclomatic complexity, V(G)
- Determine the basis path set, *e.g.*,
  - 1 - 2 - 10 - 11 - 13
  - 1 - 2 - 10 - 12 - 13
  - 1 - 2 - 3 - 10 - 11 - 13
  - 1 - 2 - 3 - 4 - 5 - 8 - 9 - 2...
  - 1 - 2 - 3 - 4 - 5 - 6 - 8 - 9 - 2...
  - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 2...
- Prepare test cases that will force execution of each basis path

Example

- value[0] = -999
- expected results: average = -999; other totals at initial values



### 21.4.1 Exemplary basis path test case

Testing Path2

- value[0] = -999
- expected results: average = -999; other totals at initial values

### 21.4.2 White-box testing method example: condition testing

- A technique that exercises **all the logical conditions**, where
- The **components** of a condition are:
  - **Simple** conditions, *i.e.*, (i) boolean variables (such as *and*, *or*) (ii) relational expressions
  - **Compound** conditions, *i.e.*, conditions composed of two or more simple conditions, boolean operators and parentheses
- Condition testing finds different types of errors, namely
  - Boolean operator errors
  - Boolean variable errors
  - Boolean parenthesis errors
  - Relational operator errors
  - Arithmetic expression error

### 21.4.3 White-box testing method example: branch testing

- Simplest condition testing technique
- For a compound condition C:
  - execute the true and false branches of C at least once
  - execute every simple condition in C at least once

## 21.5 Preparing for the topic ahead



Next lecture. . .

**Further relevant testing issues**

More specifically, we will focus on:

- Testing strategy (very important, also for exams)
- GUI testing