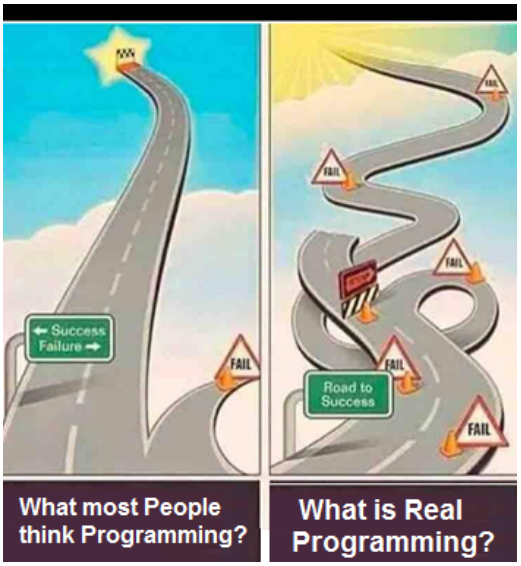
DA-IICT

IT 314: Software Engineering

Data flow and Control Flow Analysis
(White-Box Testing)

Saurabh Tiwari

1



2

Testing Strategies

Black-box testing

- Tests that validate business requirements (what the system is supposed to do).
- Test cases are derived from the requirements specification of the CUT. No knowledge of internal program structure is used.
- Also known as - functional, data-driven, or Input / Output testing.

White-box testing

- Tests that validate internal program logic (*control flow, data structures, data flow*)
- Test cases are derived by examination of the internal structure of the CUT
- Also known as - structural or logic-driven testing or clear-box, glass-box testing

Testing Strategies

▫Black box testing (Specification Based)

Equivalence Class Partitioning
▫Boundary Value Analysis
▫Cause-effect graphing
Model based Testing

White box testing (Program Based)▫

▫Control Flow Based
▫Data Flow based
Code Coverage
▫Mutation Testing

White-Box Testing

- Testing based on analysis of internal logic (design, code, etc.). (But *expected* results still come from requirements.)
- Also know as *structural testing*.
- White-box testing concerns techniques for *designing* tests; it is *not* a level of testing.
- White-box testing techniques apply *primarily* to lower levels of testing (e.g., unit and component).

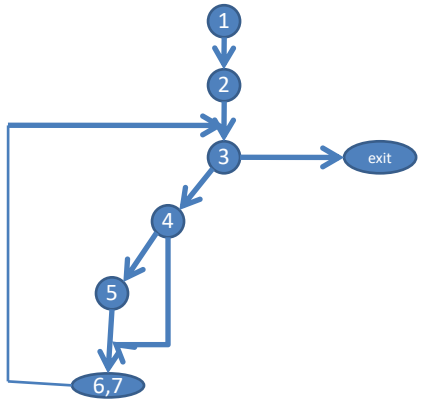
Control-flow-based Testing

- A traditional form of white-box testing
- Step 1: From the source, create a graph describing the flow of control
  - Called the control flow graph
  - The graph is created (extracted from the source code) manually or automatically
- Step 2: Design test cases to cover certain elements of this graph
  - Nodes, edges, paths

Control-flow-based Testing

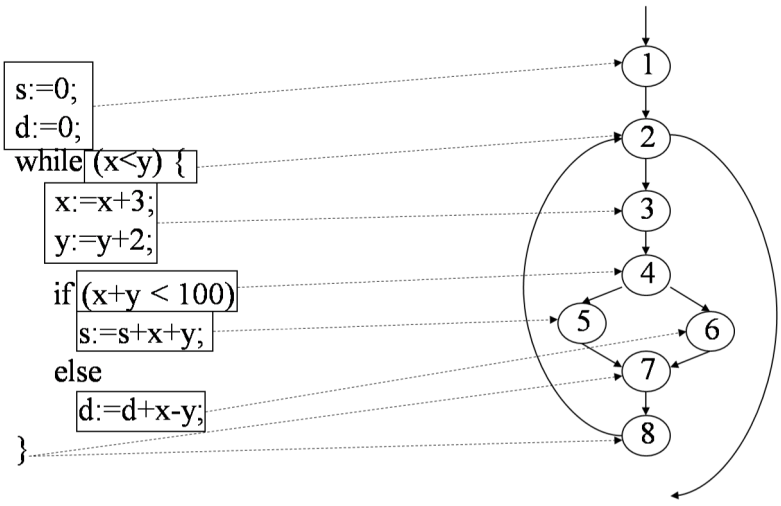
The program structure used in structural testing is the CFG i.e. control flow

```
1 read x,y;
2 z:=1;
3 while not y=0 do
  4 if x mod2 =1
      5 z:=z*x
   6 y:=y/2
   7 x:=x*x
end
```



Nodes represent Statements; Edges represent the flow

Example of a Control-flow-Graph

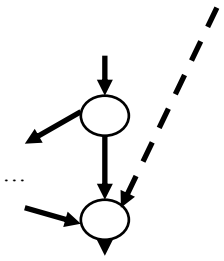


Elements of a CFG

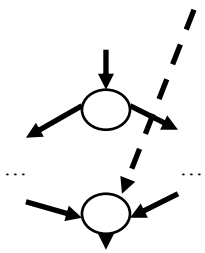
- Three kinds of nodes:
  - Statement nodes: represent single-entry-single-exit sequences of statements
  - Predicate nodes: represent conditions for branching
  - Auxiliary nodes: (optional) for easier understanding (e.g., "join points" for IF, etc.)
- Edges: represents possible flow of control
- It is relatively easy to map standard constructs from programming languages to elements of CFGs

IF-THEN, IF-THEN-ELSE, SWITCH

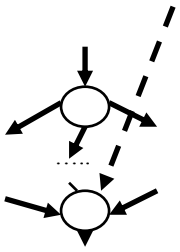
```
if (c)
   then
   // join point
```



```
if (c)
   then
else
   // join point
```

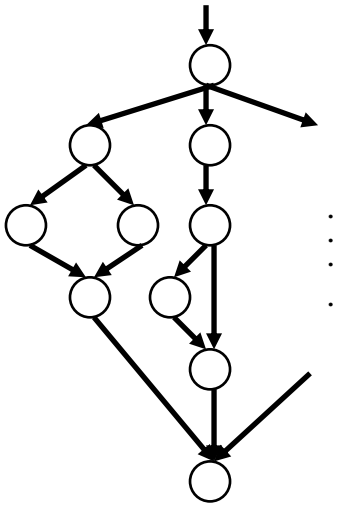


```
switch (c)
 case 1:
 case 2:
// join point
```



Example

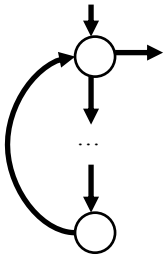
```
switch (position)
  case CASHIER
    if (empl_yrs > 5)
      bonus := 1;
    else
      bonus := 0.7;
  case MANAGER
    bonus := 1.5;
    if (retiring_soon)
      bonus := 1.2 * bonus
  case …
endswitch
```



Mapping for Loops

```
while (c) {

}
```

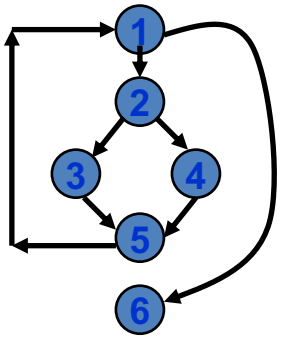


Note: other loops (e.g., FOR, DO-WHILE,…) are mapped similarly. Figure out how this is done.

Example

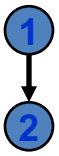
```
int f1(int x,int y){
1 while (x != y){
2    if (x>y) then
3         x=x-y;
4    else y=y-x;
5 }
6 return x;        }
```

Example Control Flow Graph



How to draw Control flow graph?

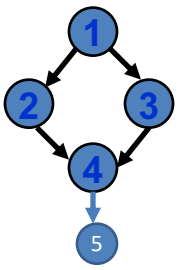
- Sequence:
  - 1 a=5;
  - 2 b=a*b-1;



15

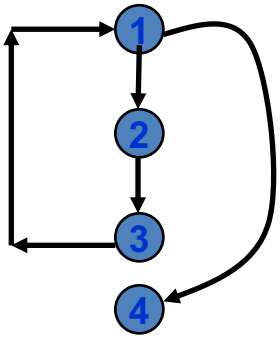
How to draw Control flow graph?

- Selection:
  - 1 if(a>b) then
  - 2 c=3;
  - 3 else c=5;
  - 5 c=c*c;



How to draw Control flow graph?

- Iteration:
  - 1 while(a>b){
  - 2 b=b*a;
  - 3 b=b-1;}
  - 4 c=b+d;



Example

```
void function eval(int a, int b, int x)
{
if (a > 1) OR (b = 0) then
    x = x /a;
if (a = 2) and (x > 1) then
    x = x + 1;
}
```

Next Lecture…
Coverage Criteria
Cyclomatic Complexity