
Globant
we are ready

UI Engineering Studio. Day 4

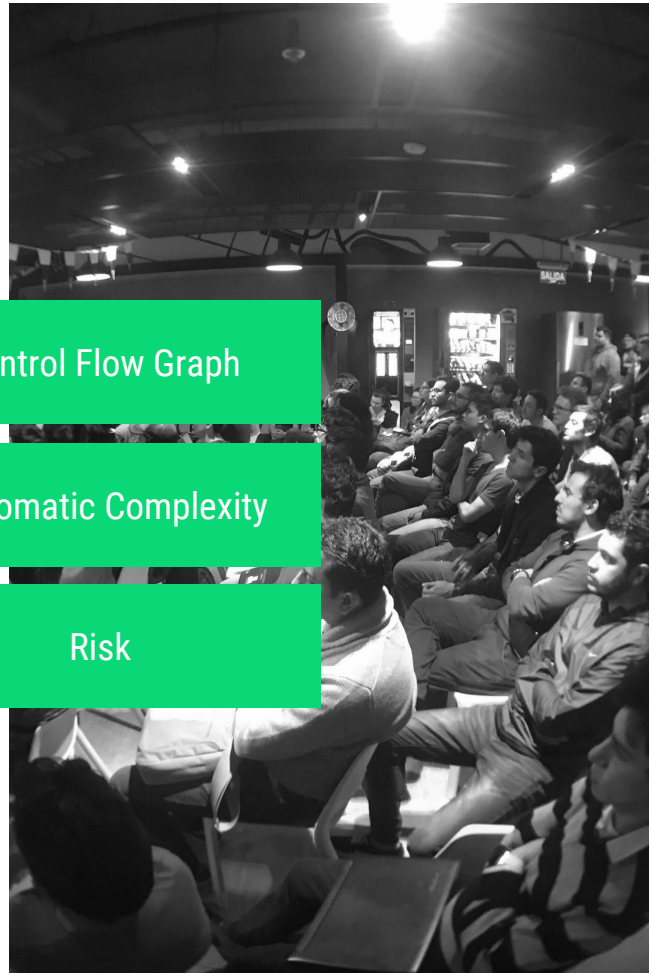
# Bootcamp: Cyclomatic Complexity

Globant

## Cyclomatic Complexity

Software metric, used to indicate the **complexity of a program**. It is a quantitative measure of the number of linearly independent paths through a program's source code.

- Wikipedia



Control Flow Graph

Cyclomatic Complexity

Risk

## UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph

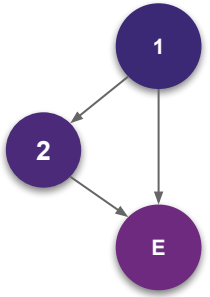
In a control flow graph each **node** in the graph represents a basic block, a straight-line piece of code without any jumps or jump targets; jump targets start a block, and jumps end a block. Directed edges are used to represent jumps in the **control flow**.

- Wikipedia

## UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph - IF

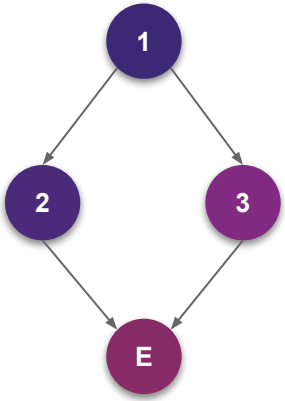
```
1   If (a) {
2       doSomething();
E   }
```



### UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph - IF ELSE

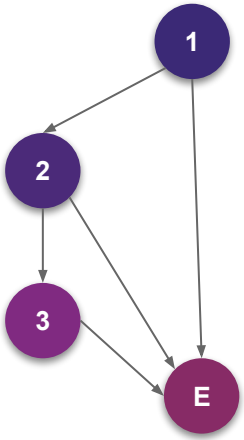
```
1   if (a) {
2       doSomething();
    } else {
3       doSomethingElse();
E   }
```



## UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph - AND

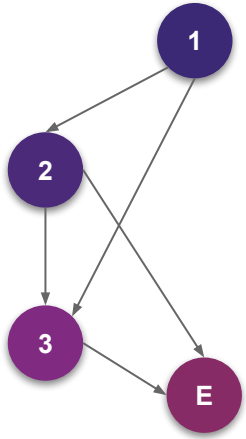
```
      1     2
    if (a && b) {
3       doSomething();
E   }
```



## UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph - OR

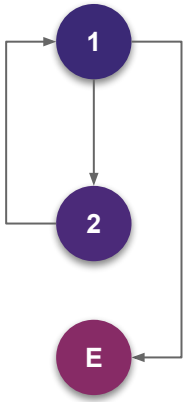
```
      1     2
    if (a || b) {
3       doSomething();
E   }
```



## UI Boot Camp: Cyclomatic Complexity

# Control Flow Graph - WHILE

```
1   While (a) {
2         doSomething();
E   }
```



UI Boot Camp

# Cyclomatic Complexity

## Avoid complexity

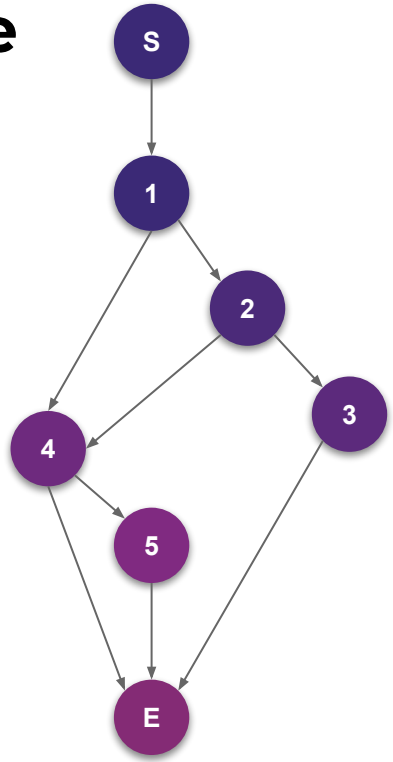
**Software Metric**, used to indicate the **complexity** of a program. It is a **quantitative measure** of the number of linearly independent paths through a program's source code.

- Wikipedia

## UI Boot Camp: Cyclomatic Complexity

# Metric - Sample code

```
S function aMethod(a) {
            1     2
        if (a && b) {
3               doSomething();
4       } else if (c){
5               doSthElse();
        }
E }
```




**M** Complexity Score

**E** number of edges of the graph.

**N** number of nodes of the graph.

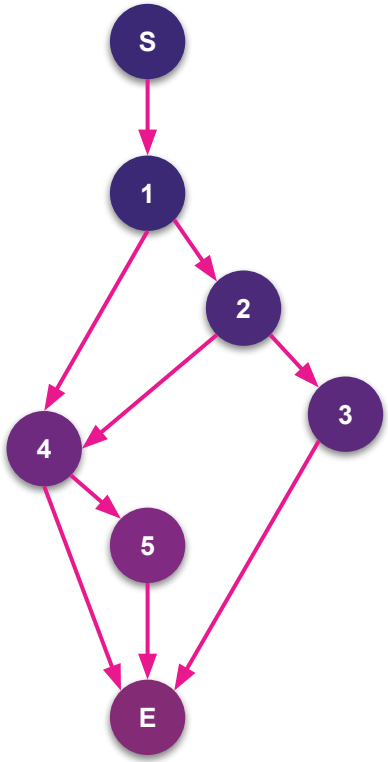
**P** number of connected components.

## UI Boot Camp: Cyclomatic Complexity

M = **Edges - Nodes** + 2

M = E - N + 2

**9 - 7 + 2 = 4**

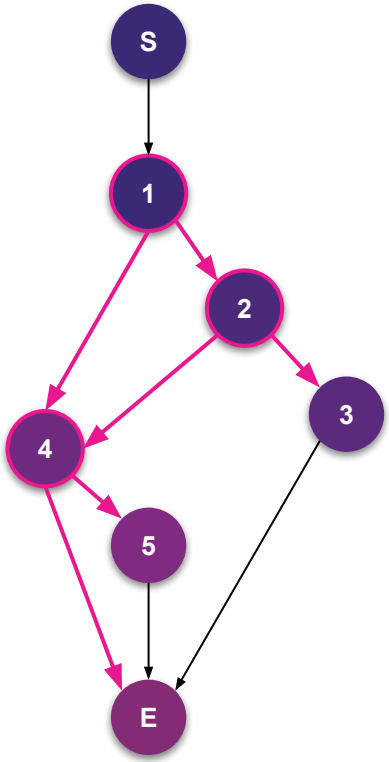


## UI Boot Camp: Cyclomatic Complexity

M = **connected components** + 1

M = P + 1

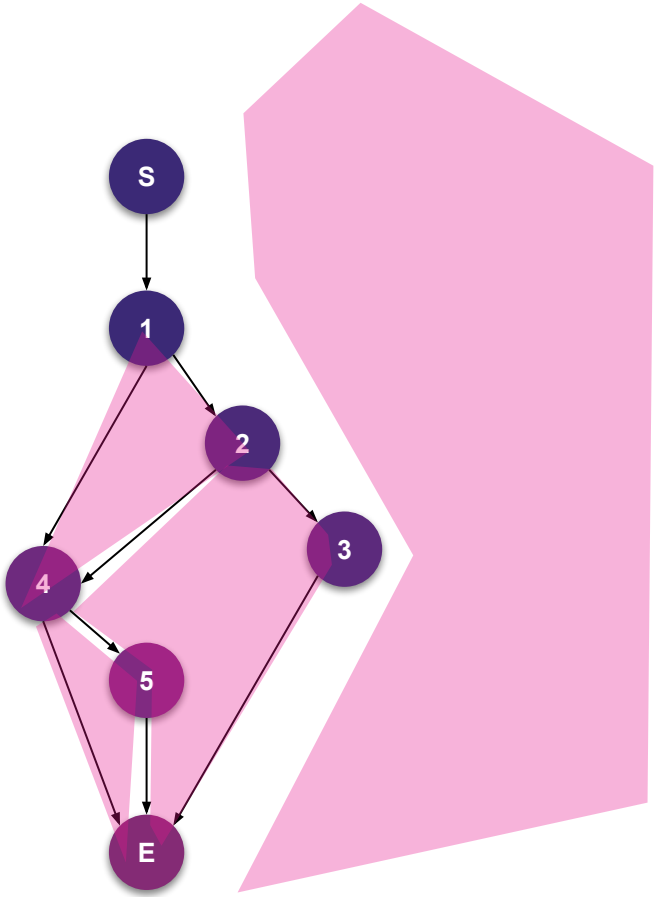
**3** + 1 = **4**



## UI Boot Camp: Cyclomatic Complexity

M = **Regions**

**4**

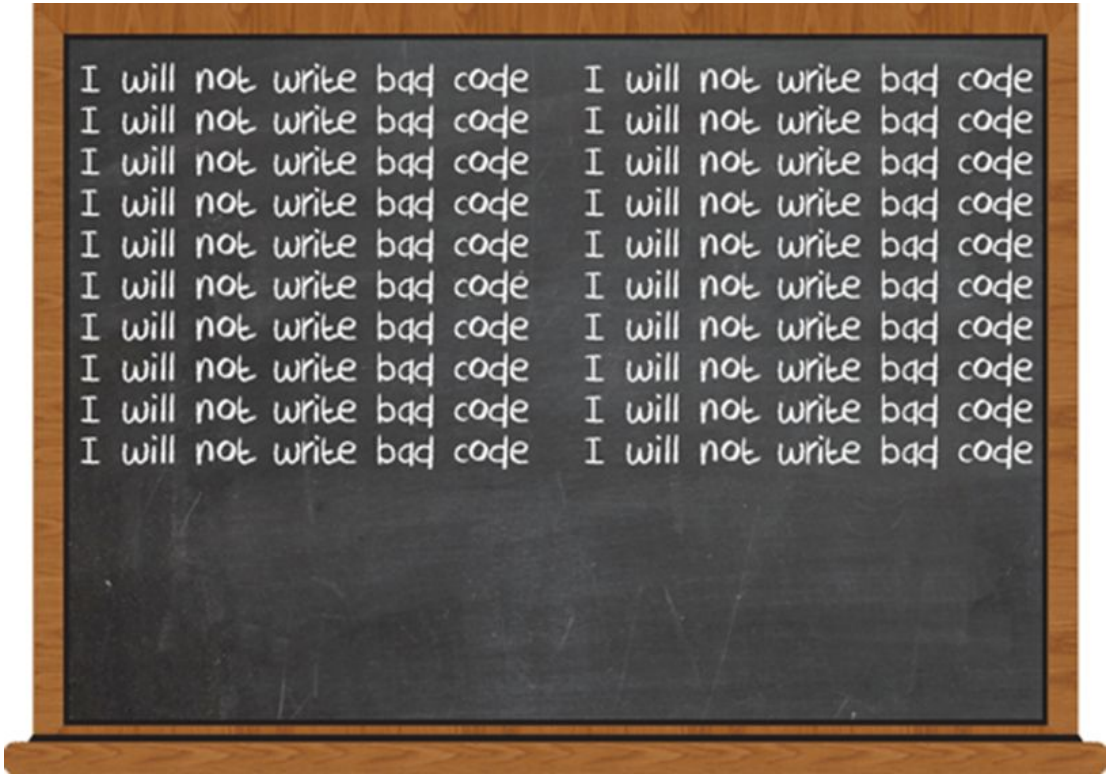


**UI Boot Camp: Cyclomatic Complexity**

# Score & Risk

| Score | Cyclomatic | Risk Type |
|---|---|---|
| 1 to 10 | Simple | Not much risk |
| 11 to 20 | Complex | Low risk |
| 21 to 50 | Too complex | Medium Risk, Attention |
| More 50 | Too complex | Can’t test, **High Risk** |

# UI Boot Camp: Cyclomatic Complexity



## UI Boot Camp: CSS

# **Homework:** Solve and calculate cyclomatic complexity

My grandfather always predicted how old people would get, and right before he passed away he revealed his secret!

In honor of my grandfather's memory we will write a function using his formula!

- Take a list of ages when each of your great-grandparent died.
- Multiply each number by itself.
- Add them all together.
- Take the square root of the result.
- Divide by two.

Example: predictAge(65, 60, 75, 55, 60, 63, 64, 45) === 86.

Note: the result should be rounded down to the nearest integer.

Code wars online **Kata**


Globant
we are ready


1
2
3
4
5
6
7
8