

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Maio de 2020

Cyclomatic Complexity

Engenharia de Segurança

Diogo Duarte PG41843

Mateus Ferreira PG37159

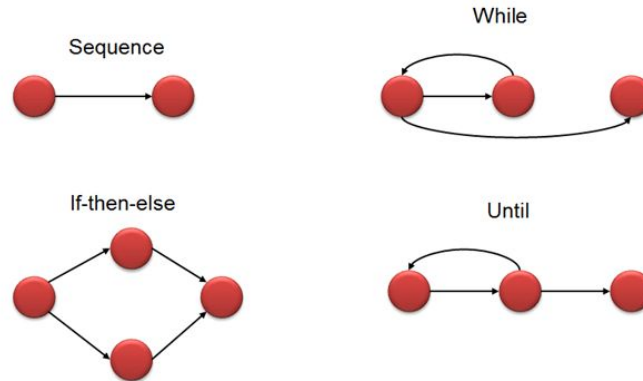
Ricardo Dias PG39295

O que é Cyclomatic Complexity

- Métrica de software para calcular complexidade do código
- Desenvolvida por Thomas J. McCabe em 1976
- Mede a quantidade de caminhos independentes que o código pode tomar
- Todo código tem pelo menos $CC = 1$

Como calcular a Cyclomatic Complexity

- Utiliza grafo de fluxo de controle
- Os nós representam as estruturas condicionais e as arestas representam os caminhos possíveis.



Como calcular a Cyclomatic Complexity

- Fórmula para calcular a Complexidade Ciclomática:

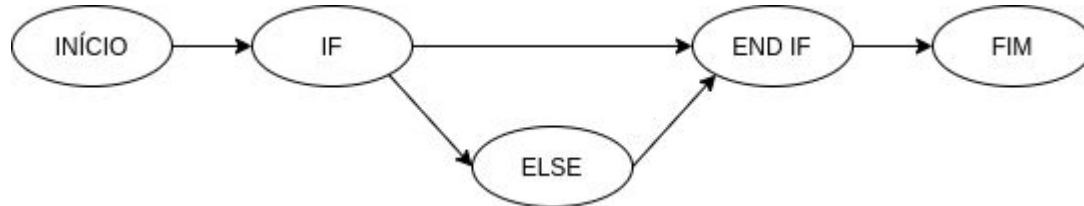
$$M = E - N + 2$$

- Onde **E** é o número de arestas e **N** é a quantidade de nós.

Como calcular a Cyclomatic Complexity

- $M = E - N + 2$
- $M = 5 - 5 + 2$
- $M = 2$

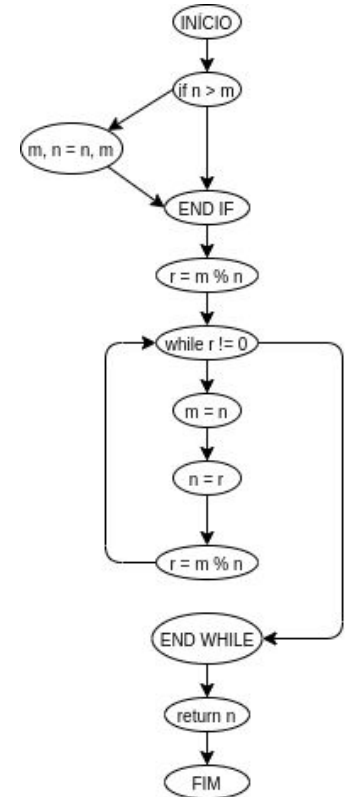
```
1 ▼ def age(i):  
2   ▼  
3     Retorna se o número inserido  
4     é maior ou igual a 18.  
5     :param i:  
6         número a ser verificado  
7     :return:  
8         True se o número for maior que 18 ou False se menor.  
9     "" ""  
10  ▼ if i >= 18:  
11      return True  
12  ▼ else:  
13      return False  
14
```




Como calcular a Cyclomatic Complexity

- $M = E - N + 2$
- $M = 13 - 12 + 2$
- $M = 3$

```
1 ▼ def euclid(m, n):  
2   ▼  
3     """  
4     Algoritmo de Euclides  
5     Assumindo que m e n são maiores que 0,  
6     retorna o Máximo Divisor Comum entre eles.  
7     :param m:  
8         Número inteiro maior que 0  
9     :param n:  
10        Número inteiro maior que 0  
11    :return:  
12        Máximo divisor comum entre m e n  
13    """  
14    ▼ if n > m:  
15        m, n = n, m  
16        r = m % n  
17    ▼ while r != 0:  
18        m = n  
19        n = r  
20        r = m % n  
21    return n  
22
```



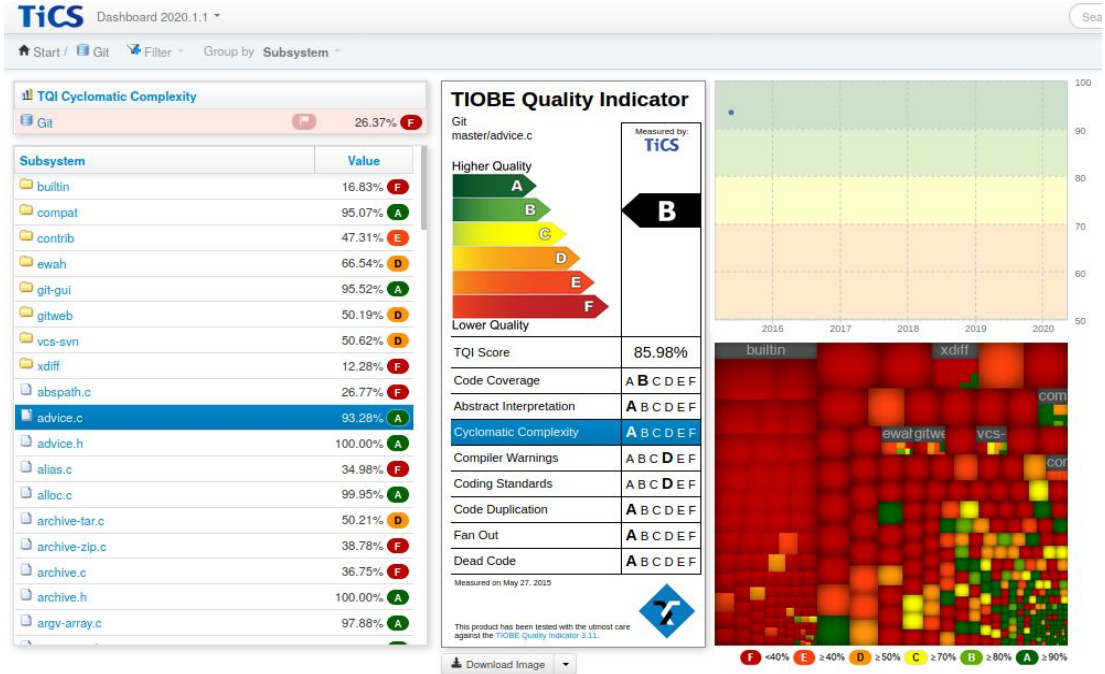
Ferramentas para a medição da Complexidade Ciclomática

- TICSpp
 - Mlint e Checkcode
 - SonarQuebe
 - Eclipse Metrics Plugin
 - JaCoCo Jenkins pipeline plugin
 - Kuscos Application
- 
- Lizard

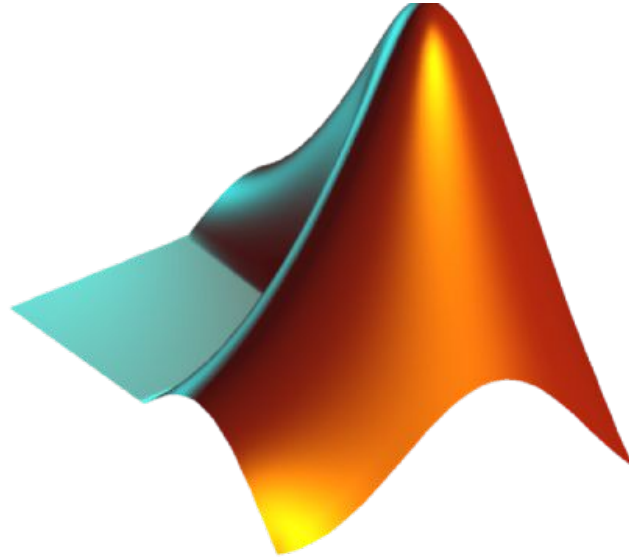
TICSpP



Cyclomatic Complexity	TQI Score	Category
<= 2.57	>= 90%	A
<= 3.00	>= 80%	B
<= 3.42	>= 70%	C
<= 4.37	>= 50%	D
<= 5.00	>= 40%	E
> 5.00	< 40%	F



Mlint e Checkcode



MATLAB R2019a

JaCoCo Jenkins pipeline plugin

JaCoCo

Kuscos Application



SonarQube



Eclipse Metrics Plugin



Lizard

C/C++	Ruby
Java	TTCN-3
C#	PHP
JavaScript	Scala
Objective-C	GScript
Swift	Golang
Python	Rust

```
[*]-[mateus@parrot]-[~/Downloads]
$ lizard wadread.c
```

NLOC	CCN	token	PARAM	length	location
8	1	45	1	8	SwapLONG@107-114@wadread.c
5	1	23	1	5	SwapSHORT@116-120@wadread.c
5	1	23	1	5	derror@127-131@wadread.c
5	2	23	1	5	strupr@134-138@wadread.c
7	2	36	1	9	filelength@140-148@wadread.c
31	4	239	1	44	openwad@152-195@wadread.c
24	4	135	2	31	loadlump@198-228@wadread.c
19	2	135	2	26	getsfx@231-256@wadread.c

```
1 file analyzed.
```

NLOC	Avg.NLOC	Avg.CCN	Avg.token	function_cnt	file
140	13.0	2.1	82.4	8	wadread.c

```
No thresholds exceeded (cyclomatic_complexity > 15 or nloc > 1000000 or length > 1000 or parameter_count > 100)
```

Total nloc	Avg.NLOC	Avg.CCN	Avg.token	Fun Cnt	Warning cnt	Fun Rt	nloc Rt
140	13.0	2.1	82.4	8	0	0.00	0.00

Como melhorar a Complexidade Ciclomática

A complexidade desnecessária é um entrave para testes eficazes por três razões

- a lógica extra requer mais testes
- os testes para esta complexidade extra requerem um esforço extra para executar cada um
- por vezes pode ser impossível testar a lógica extra devido às dependências de controlo de fluxo do software

Como melhorar a Complexidade Ciclomática

Complexidade atual e Complexidade realizável:

- Complexidade atual (ac) é definida pelo número de caminhos linearmente independentes que foram executados durante o teste
- Complexidade realizável (rc) é a complexidade real máxima possível, ou seja, a contagem do conjunto de caminhos induzidos por todos os testes possíveis

Como melhorar a Complexidade Ciclomática

Remover dependências de controlo:

- Permitem uma melhoria do código pois os módulos tendem a ser menos complexos e com uma lógica de decisão direta.

Como melhorar a Complexidade Ciclomática

Trade-offs de reduzir a complexidade

- Lógica não estruturada
- Middle-ground

Aplicações

1. Limitar a complexidade durante o desenvolvimento
2. Medir a "estrutura" de um programa
3. Implicações para testes de software
4. Correlação com o número de defeitos

Conclusão

- Importância de utilizar métricas no desenvolvimento de software.
- Relação entre aumento de funcionalidades e aumento de linhas de código.
- Aumento do número de linhas leva a um aumento de bugs, alguns destes originam vulnerabilidades.
- Verifica-se que a complexidade ciclomática influencia a segurança de software