Software Metrics

João Saraiva

HASLab / INESC TEC & Universidade do Minho

2020/2021

Software Metrics

"Not everything that counts can be counted, and not everything that can be counted counts."

Albert Einstein

Software Metrics

Porquê medir Software?

Compreender questões relacionadas com o desenvolvimento do Software.

- Tomar decisões sobre esse desenvolvimento com base em factos e não apenas opiniões.

- Prever condições para desenvolvimentos futuros.

Software Metrics

Porquê medir Software?

Para estimar o custo e o tempo de desenvolvimento.

. Para melhorar a qualidade do software.

Software Metrics

O que medir num sistema Software?

Tempo de execução do programa!

. Consumo de Memória.

. Consumo de Energia.

Usabilidade do sistema de software.

Software Metrics

LOC: Number of Lines of Code

. A métrica mais simples e mais usada para "medir" o código fonte de um programa.

. Fácil de calcular e de automatizar!

Software Metrics

LOC: Number of Lines of Code

Se eu disser que o meu programa tem 400 linha de código. O que posso eu dizer sobre ele?

. Hum

. E se eu disser ainda que o programa contém 200 métodos?

Software Metrics: LOC

De facto, é muito usada como uma medida de normalização:

- effort/cost estimation

  Effort = f(LOC)

- quality assessment/estimation

  Qualidade = defects/LOC

- productivity assessment

  Produtividade = LOC/effort

Software Metrics: LOC

Outras métricas equivalentes:

KLOC: Thousands of Lines Of Code
KDSI: Thousands of Delivered Source Instructions
NCLOC: Non-Comment Lines of Code

- Number of Characters or Number of Bytes

```c
/*
    Análise e Teste de Software
                LOC
*/
#include <stdio.h>
#ifdef ATS
#include "ats.h"
#endif

int main()
{
   int i,aux;
   scanf("%d",&aux);
   for (i=0 ; i < 10 ; i++)
     { printf("%d : %d\n",i,aux);
     }
   return 0;
}
```

```
> wc -l metricas.c
18 metricas.c
```

Software Metrics

Contar Linhas de Código:

Devem-se contar as linhas de código reutilizadas (sem alterações)?

Devem-se contar as linhas de código não executadas? (comentários, macros, etc)

E as Diretivas de compilação?

E linhas em branco?

E linhas só com “sinais de pontuação”?

Software Metrics: SLOCCount

Contar Linhas de Código:

SLOCCount: contém um conjunto de programas para contar linhas de código (SLOC) para grandes sistemas de software.

.Linux: apt install sloccount

Software Metrics

SLOCCount suporta a maior parte das linguagens de programação diferentes.

```
SLOC    Directory       SLOC-by-Language (Sorted)
24728   src_modules     ansic=24728
19067   src_main        ansic=19067
8011    src_lib         ansic=8011
5501    src_os          ansic=5340,sh=106,cpp=55
3886    src_support     ansic=2046,perl=1712,sh=128
3823    src_top_dir     sh=3812,ansic=11
3788    src_include     ansic=3788
3469    src_regex       ansic=3407,sh=62
2783    src_ap          ansic=2783
1378    src_helpers     sh=1345,perl=23,ansic=10
1304    top_dir         sh=1304
104     htdocs          perl=104
31      cgi-bin         sh=24,perl=7
0       icons           (none)
0       conf            (none)
0       logs            (none)


ansic:          69191 (88.85%)
sh:              6781 (8.71%)
perl:            1846 (2.37%)
cpp:               55 (0.07%)


Total Physical Source Lines of Code (SLOC)                     = 77873
Estimated Development Effort in Person-Years (Person-Months) = 19.36 (232.36)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Estimated Schedule in Years (Months)                           = 1.65 (19.82)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers  (Effort/Schedule)      = 11.72
Total Estimated Cost to Develop                                = $ 2615760
 (average salary = $56286/year, overhead = 2.4).
```

Software Metrics: SLOCCount

http://www.dwheeler.com/sloccount/sloccount.ht

```
1  /*
2      Análise
3
4  */
5  #include <st
6  #ifdef ATS
7  #include "at
8  #endif
9
10 int main()
11 {
12   int i,aux;
13   scanf("%d'
14   for (i=0 ;
15     { printf( %d . %d\n",i,aux);
16     }
17   return 0;
18 }
```

```
SLOC    Directory       SLOC-by-Language (Sorted)
13      top_dir         ansic=13


Totals grouped by language (dominant language first):
ansic:           13 (100.00%)



Total Physical Source Lines of Code (SLOC)                = 13
Development Effort Estimate, Person-Years (Person-Months) = 0.00 (0.03)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.05 (0.62)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 0.04
Total Estimated Cost to Develop                           = $ 283
 (average salary = $56,286/year, overhead = 2.40).
SLOCCount, Copyright (C) 2001-2004 David A. Wheeler
```

Software Metrics

Problemas com as métricas baseadas em LOC:

- Não há uma definição standard!
- Mede o comprimento do programa e não o seu "tamanho"
- Erradamente usadas para definir:
  - esforço
  - complexidade
  - funcionalidade
- Não consideram redundância e reuso de código!
- Não permitem comparações entre programas escritos em diferentes linguagens
- Disponíveis apenas no fim do desenvolvimento.

Software Metrics: SLOCCount

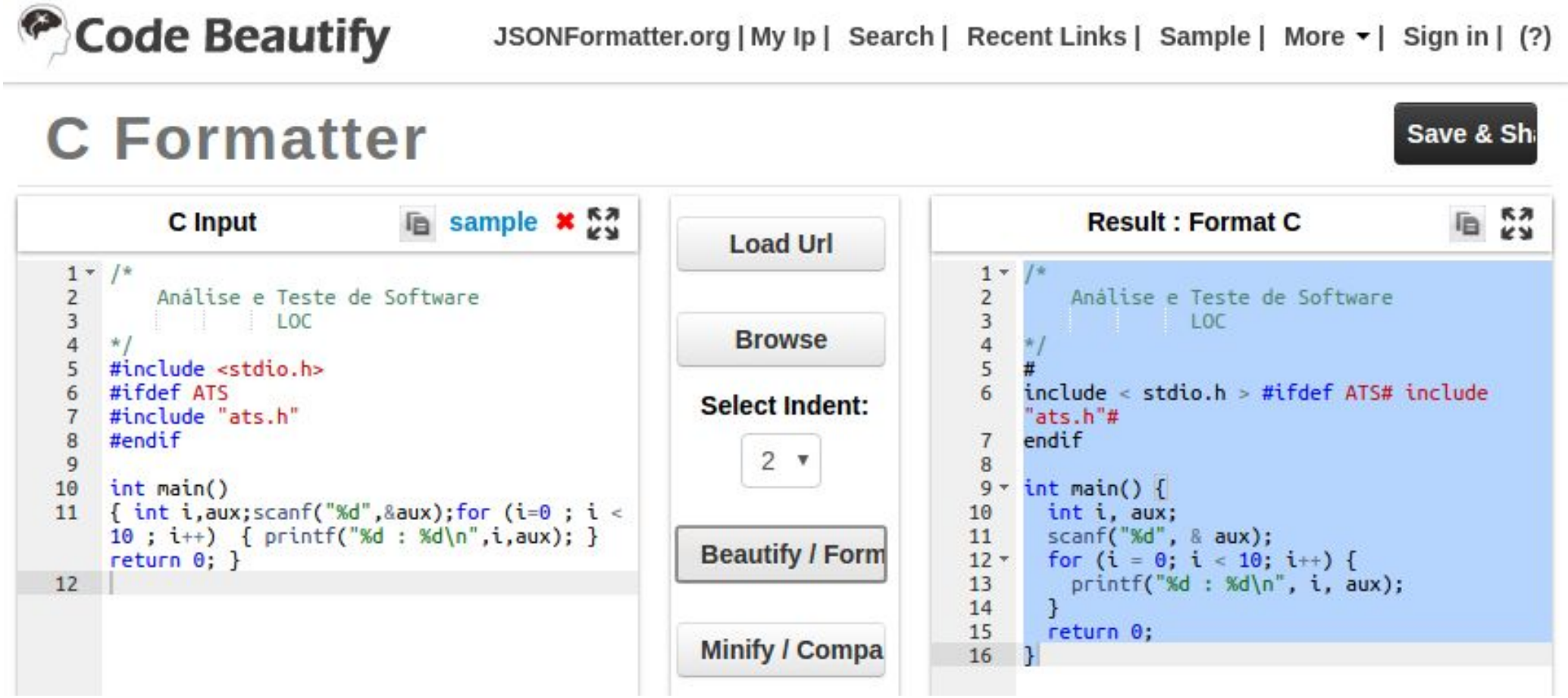
```c
/*
    Análise e Teste de Software
              LOC
*/
#include <stdio.h>
#ifdef ATS
#include "ats.h"
#endif

int main()
{ int i,aux;scanf("%d",&aux);for (i=0 ; i < 10 ; i++)  { printf("%d : %d\n",i,aux
x); } return 0; }
```

Quantas Linhas?

```
SLOC    Directory       SLOC-by-Language (Sorted)
6       top_dir         ansic=6


Totals grouped by language (dominant language first):
ansic:             6 (100.00%)




Total Physical Source Lines of Code (SLOC)                = 6
Development Effort Estimate, Person-Years (Person-Months) = 0.00 (0.01)
 (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
Schedule Estimate, Years (Months)                         = 0.04 (0.45)
 (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
Estimated Average Number of Developers (Effort/Schedule)  = 0.02
Total Estimated Cost to Develop                           = $ 126
 (average salary = $56,286/year, overhead = 2.40).
```

Software Metrics: SLOCCount

Possible solution: **Before counting we consider a common layout!**



Software Metrics

Source Code Metrics:

Syntactic Metrics
- LOC – lines of code
- NOF – number of functions/methods
- NOA – number of arguments of a fun
- number of classes/methods, etc

Complexity Metrics
- McCabe Complexity Metric
- Halstead suit

Documentation Metrics
- Number of comments

Software Metrics

McCabe Complexity Metric (or *cyclomatic complexity*) of a fragment of a program's source code is the count of the number of linearly independent paths through the source code.

Cyclomatic Complexity

Cyclomatic Complexity (CC) is computed using the *control flow graph* of the program: a directed graph containing the basic blocks of the program, with an edge between two basic blocks if control may pass from the first to the second. CC is then defined as:

$$CC = E - N + 2P$$

where

E = the number of edges of the graph

N = the number of nodes of the graph

P = the number of connected components

Cyclomatic Complexity

Cyclomatic Complexity (CC(N))

McCabe, 1976

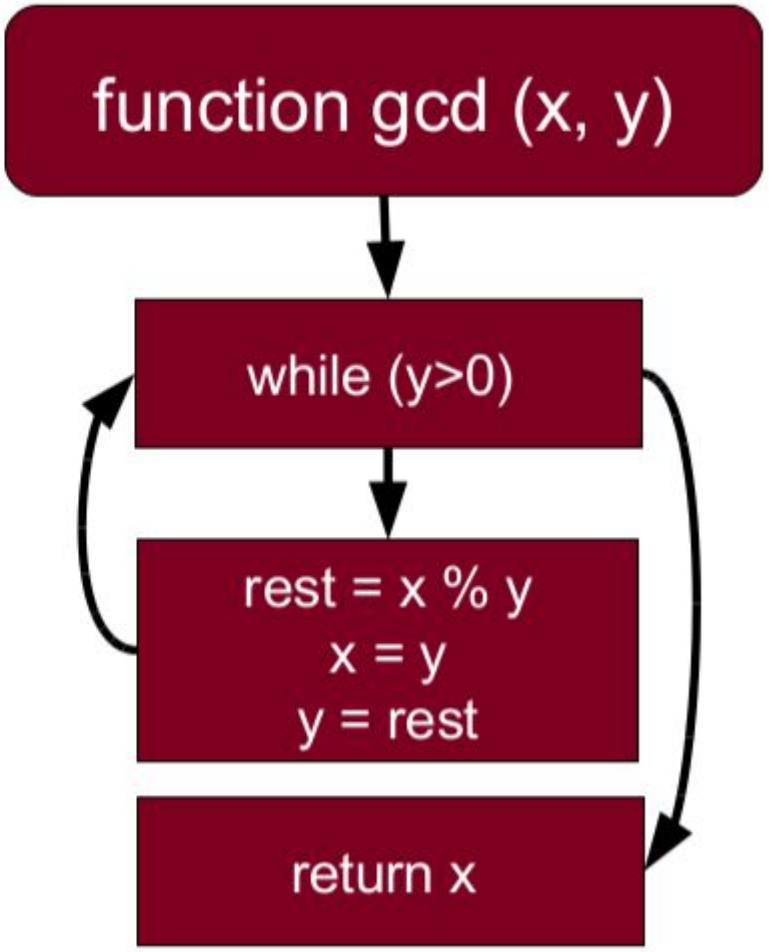
$$CC = e - n + 2\mathrm{p}$$

p: number of components

e: number of edges

n: number of nodes

Example CC = 4 – 4 + 2*1 = 2



Cyclomatic Complexity

Original McCabe values	PMD (per method)
1-10 : low complexity	1-4 : low complexity
11-20 : medium complexity	5-7 : medium complexity
21-50 : high complexity	8-10 : high complexity
> 51 : very high complexity	> 11 : very high complexity

Cyclomatic Complexity

Cyclomatic complexity is

Is not only useful for accessing the complexity of a program.

It can be used to define the test coverage of a program, i.e., the number of test cases.

Halstead Complexity Suit

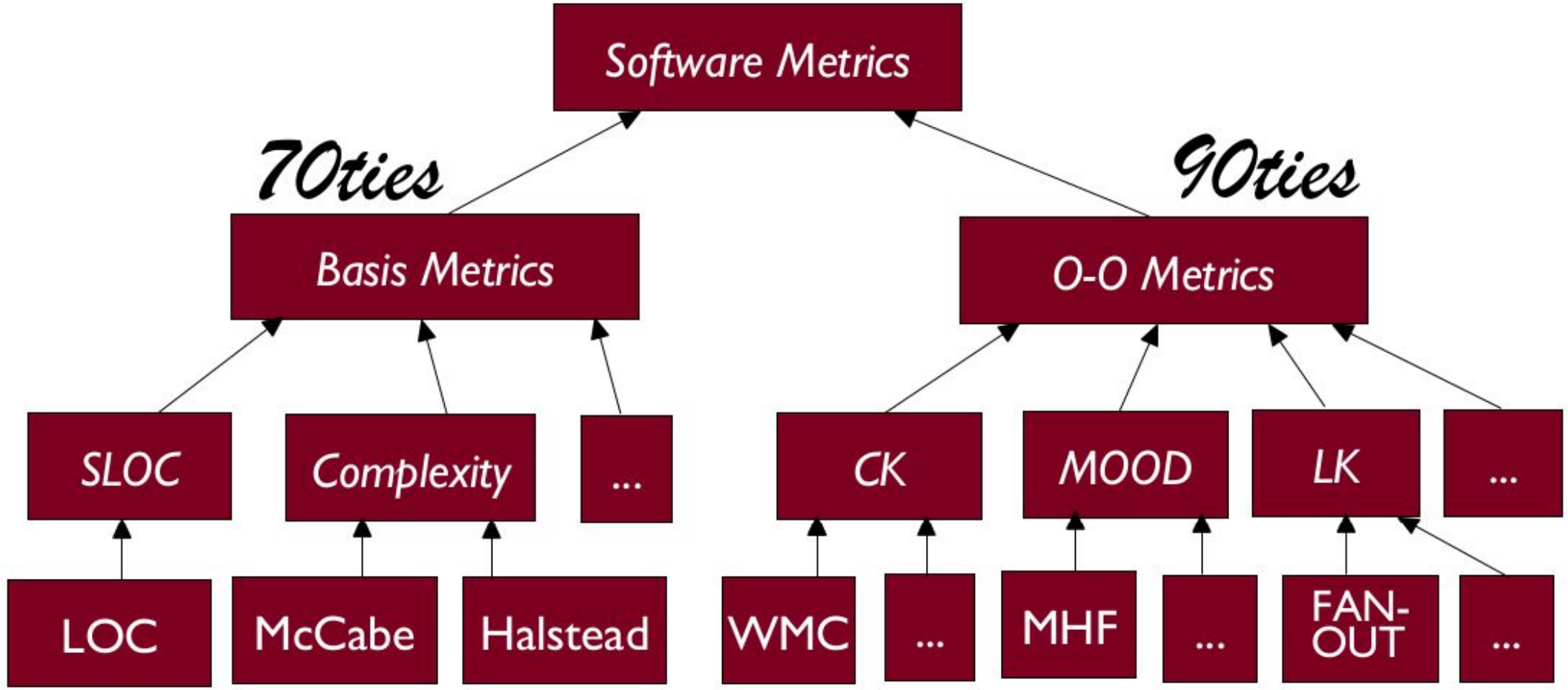
Let

n1 – number of distinct operators
n2 – number of distinct operands
N1 – total number of operators
N2 – total number of operands

in

Prog. Vocabulary: n = n1 + n2
Prog. length : N = N1 + N2
Volume : V = N * log_2 n
Difficulty : D = n1/2 + N2/n2
Effort : E = V * D

Software Metrics

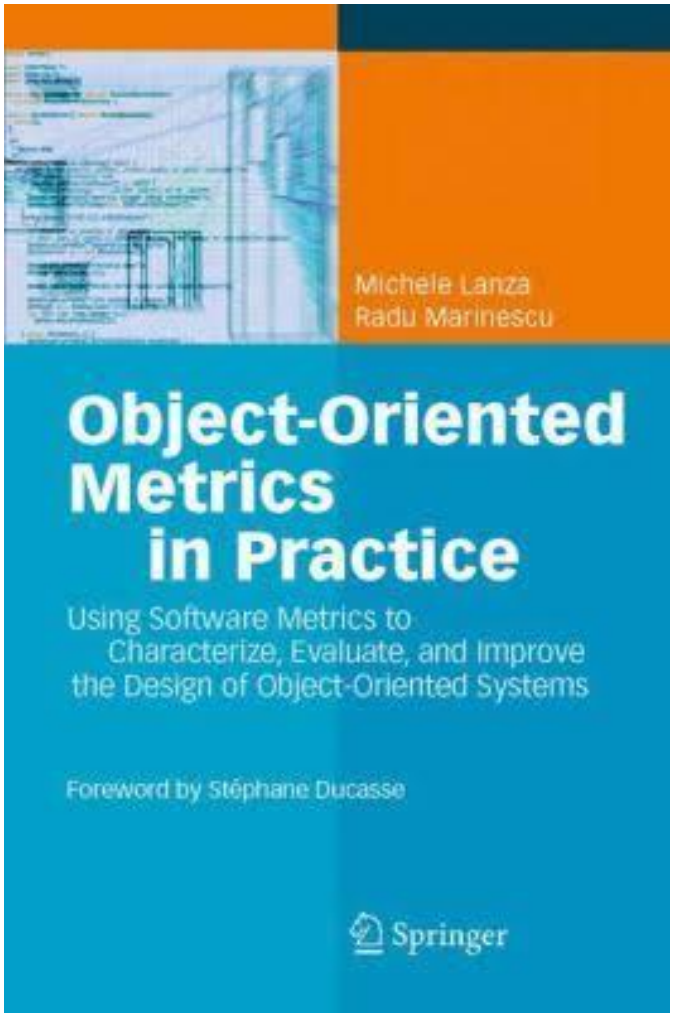


Software Metrics

Book:

Object-Oriented Metrics in Practice:

Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems



Software Metrics: Visualization

Software Metrics: Visualization

Www.d3js.org

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.