




Métricas e Smells para C-- e MSP

Métodos Formais em Engenharia de Software
Análise e Teste de Software
2015/2016

Cláudia Ribeiro A64460
José Ribeiro A64389
Mário Santos A64299

Conteúdo

- ❏ Sistema
 - ❏ Métricas
 - ❏ Bad Smells
 - ❏ Star Ranking
 - ❏ Refactoring
 - ❏ Conclusão
- 

Métricas e Smells para C++ e MSP

Definição de um catálogo de métricas de código fonte para a linguagem C++;

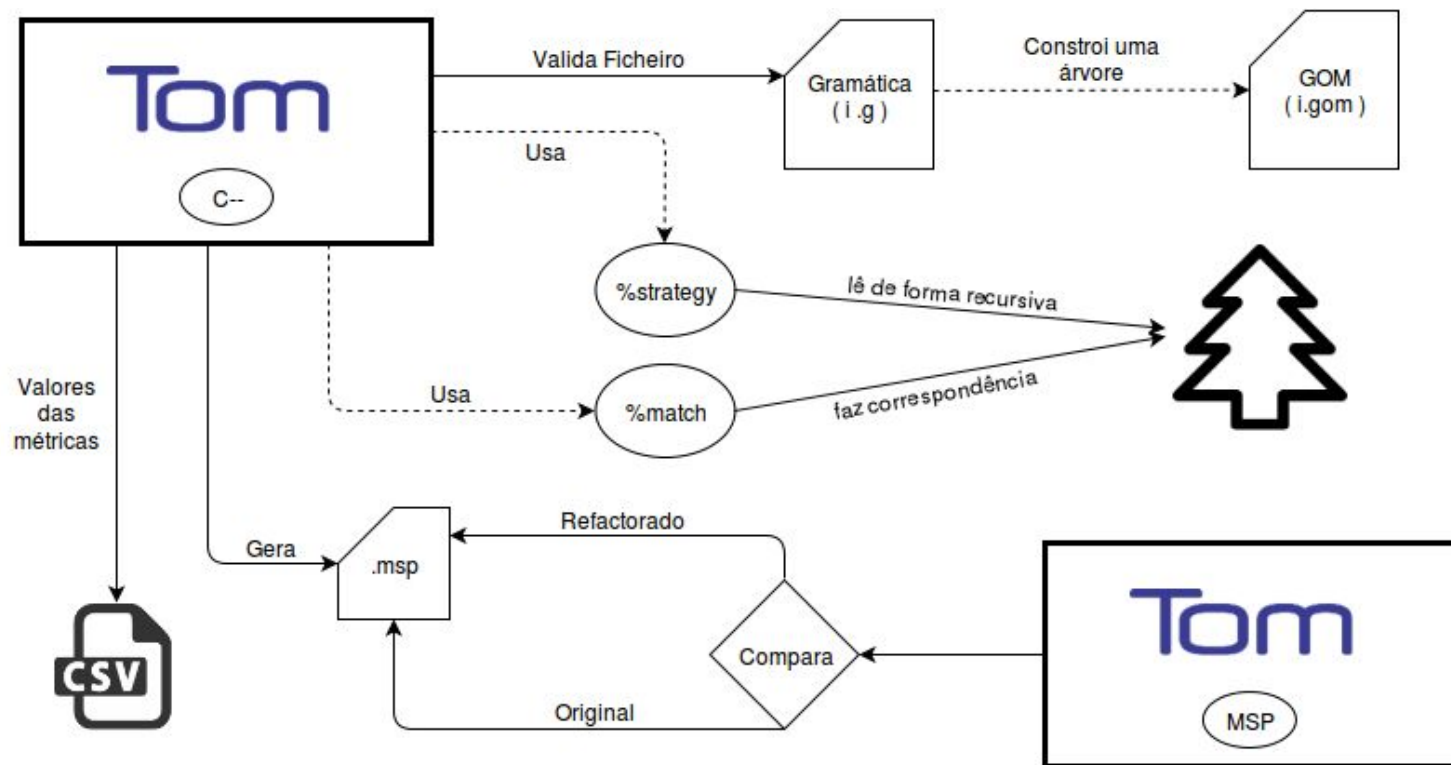
Definição de "bad smells" em C++ em função de valores dessas métricas;

Aplicação de "refactoring" onde são localizados "bad smells", eliminando automaticamente esse pedaço de código;

No final, são executados ambos os programas .msp (original e o refabricado) e são feitas comparações os seus valores.



Sistema



Métricas

- ❑ LOC (Lines of Code)
- ❑ NOD (Number of Declarations)
- ❑ NOA (Number of Arguments)
- ❑ NBD (Nested Block Depth)
- ❑ CC (Cyclomatic Complexity)

Para as métricas em MSP, apenas foram criados contadores para cada variável que o ficheiro .msp gerado contém.



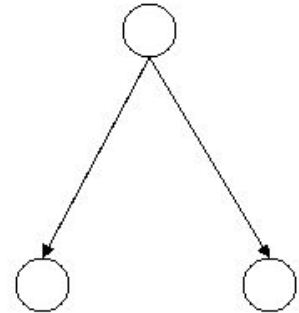
Métricas - Cyclomatic Complexity

```
/*vai calcular o cyclomatic complexity com a formula D+1 (D = pontos de decisão)*/
private static int foundCC(Instrucao i) {
    %match(i) {
        If(condicao,inst1,inst2) -> { return 1+foundCC(`inst1)+foundCC(`inst2)+foundBoolean(`condicao);}
        While(condicao,inst) -> { return 1+foundCC(`inst)+foundBoolean(`condicao);}
        For(decl,condicao,exp,inst) -> { return 1+foundCC(`inst)+foundBoolean(`condicao);}
        SeqInstrucao(inst1, inst*) -> { return foundCC(`inst1)+foundCC(`inst*);}
    }
    return 0;
}

/* Numero de operações booleanas nas operações de condição (auxiliar para o CC) */
private static int foundBoolean(Expressao e) {
    %match(e) {
        E(cond1,cond2) -> { return 1+foundBoolean(`cond1)+foundBoolean(`cond2);}
        Ou(cond1,cond2) -> { return 1+foundBoolean(`cond1)+foundBoolean(`cond2);}
    }
    return 0;
}
```



Sequential flow



Decision flow

Bad Smells

- ❑ Os limites para a implementação do *Star Ranking* são recolhidos através de um ficheiro .csv (Excel).
- ❑ Caso o limite seja ultrapassado ao fazer o *ranking*, o programa alerta que detectou um “bad smell”.

```
----> Funcao: main  
Numero de Linhas: 19 | Max(15)  
** Smell Detectado: Metodo Extenso **  
  
Numero de Declaracoes: 3 | Max(5)  
Numero de Argumentos: 1 | Max(3)  
Maior Bloco Aninhado: 3 | Max(3)  
Cyclomatic Complexity: 5 | Max(5)  
-> Star Ranking: 4,8 em 5
```

Star Ranking

A partir da classificação das métricas de um dado ficheiro podemos obter os seguintes valores máximos, sendo que na fórmula final o pior programa terá 1 valor e o melhor terá 5 estrelas.

- ❑ Número de Linhas: 0,5 valores
- ❑ Número de Declarações: 0,5 valores
- ❑ Número de Argumentos: 1 valor
- ❑ Número de Blocos Aninhados: 1,5 valores
- ❑ *Cyclomatic Complexity*: 1,5 valores



Refactoring

A partir de alguns “bad smells” detectados é necessário transformar algum código que poderá estar a não ser utilizado ou variáveis que não façam a diferença no desenrolar do programa.

→ Variáveis:

- ❑ Remoção de argumentos não utilizados;
- ❑ Remoção de declarações não utilizadas;
- ❑ Remoção de parâmetros não utilizados.

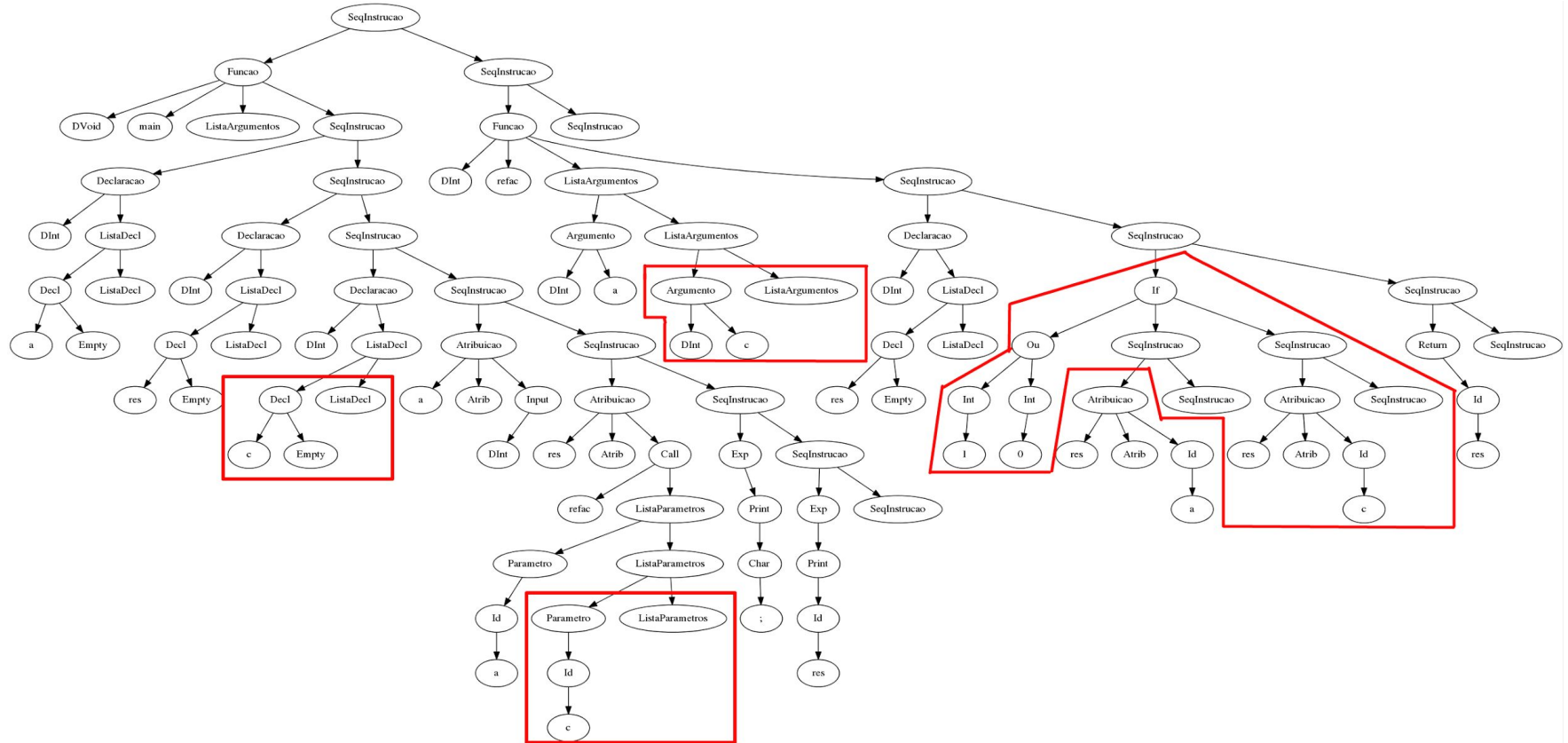
→ Outros:

- ❑ Remoção de *Dead Code*;
- ❑ Remoção da negações de condições.

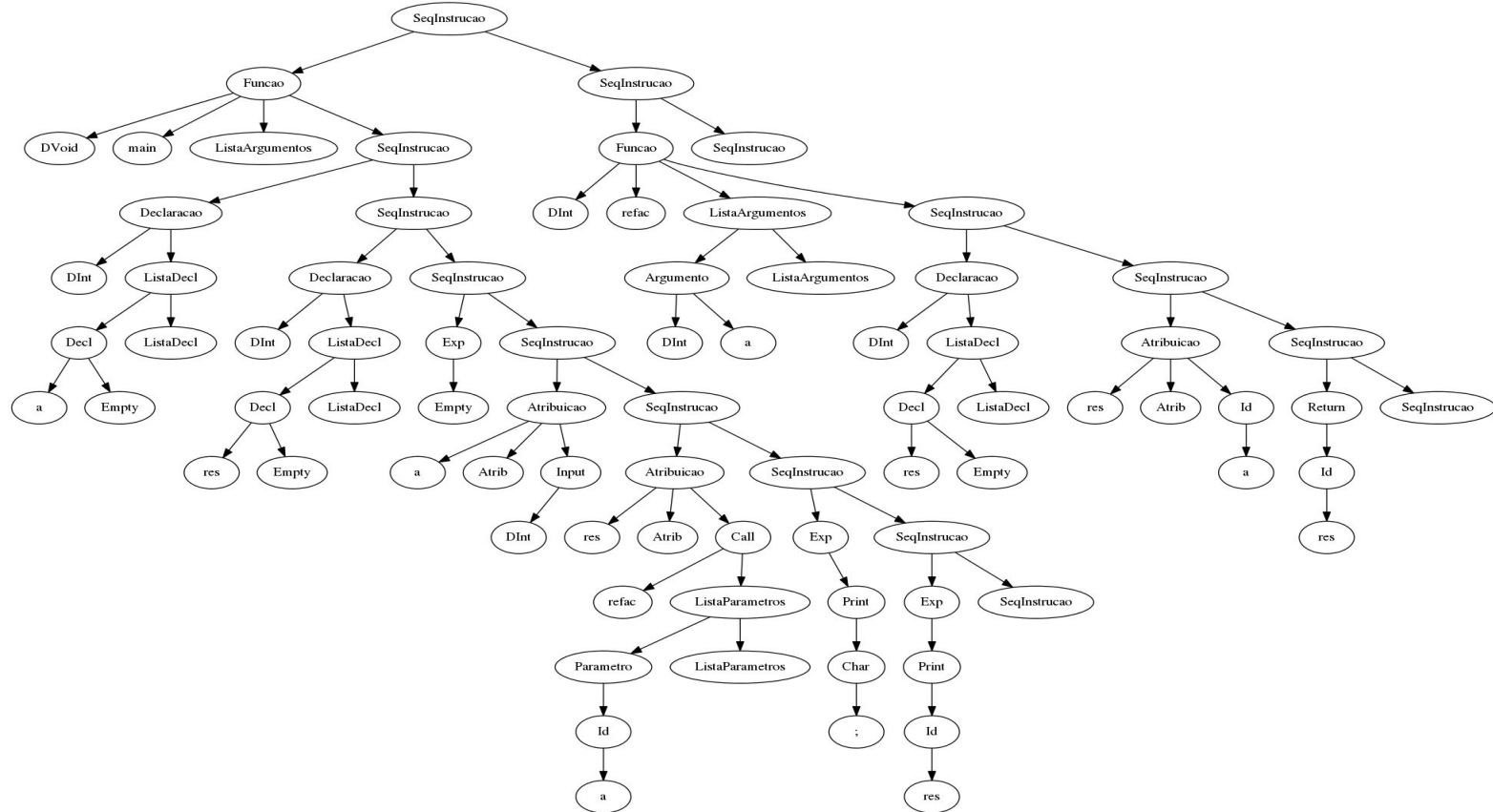
Apenas são modificados ficheiros C - -.



Refactoring



Refactoring



Refactoring

```
1 void main() {
2     int a;
3     int res;
4     int c;
5     a = input(int);
6     res = refac(a,c);
7     print(';');
8     print(res);
9 }
10
11 int refac(int a, int c){
12     int res;
13     if (1 || 0) {
14         res = a;
15     }
16     else {
17         res = c;
18     }
19     return res;
20 }
```

| Instrução | Original | Refactored |
|-----------|----------|------------|
| ALabel | 4 | 2 |
| Mod | 0 | 0 |
| Store | 7 | 5 |
| Load | 7 | 5 |
| Decl | 8 | 6 |
| Halt | 1 | 1 |
| Jump | 1 | 0 |
| Pusha | 14 | 10 |
| Push | 5 | 3 |
| Gt | 0 | 0 |
| GoEq | 0 | 0 |
| Eq | 0 | 0 |
| Mul | 0 | 0 |
| Add | 0 | 0 |
| Dec | 0 | 0 |
| Nott | 0 | 0 |
| Sub | 0 | 0 |
| IIn | 1 | 1 |
| Jumpf | 1 | 0 |
| Neq | 0 | 0 |
| Ret | 1 | 1 |
| Lt | 0 | 0 |
| Call | 1 | 1 |
| Inc | 0 | 0 |
| PushA | 0 | 0 |
| IOut | 4 | 4 |
| Or | 1 | 0 |
| And | 0 | 0 |
| Div | 0 | 0 |
| LoEq | 0 | 0 |

```
1 void main() {
2     int a;
3     int res;
4     a = input(int);
5     res = refac(a);
6     print(';');
7     print(res);
8 }
9
10 int refac(int a){
11     int res;
12     res = a;
13     return res;
14 }
```

Conclusão

- ❑ Uma primeira abordagem de forma errada;
- ❑ Pouco à vontade com o TOM que foi ultrapassado com o tempo;
- ❑ Refactoring (TOM + GOM);
- ❑ Compromissos cumpridos.





Métricas e Smells para C-- e MSP

Métodos Formais em Engenharia de Software
Análise e Teste de Software
2015/2016

Cláudia Ribeiro A64460
José Ribeiro A64389
Mário Santos A64299