Trabalhos Práticos - TP2

Arquitetura de Software - 2018/2

Alberane Lúcio Thiago da Cunha - alberane.cunha@posgrad.ufla.br
Willian Gaudêncio Rezende - willian.rezende@posgrad.ufla.br

Problema

- Fornecer métricas sobre o código em análise
  - Complexidade Ciclomática (CCV)
  - Número de Atributos (NOA)
  - Número de funções de Acesso a Disco (AFS)
- Porque devemos ficar atento a códigos onde esses indicadores são mais elevados?
  - Sistemas **complexos** e que fazem acesso a **disco** podem ser tratados como **críticos**, já que falhas podem trazer **problemas graves** ao **sistema de arquivos**

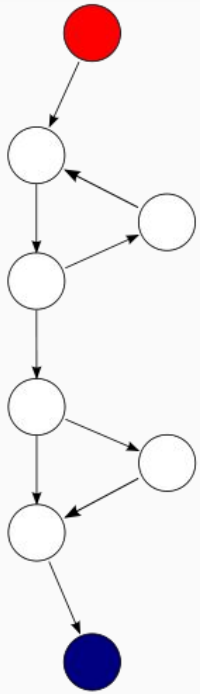
Sobre as métricas analisadas...

CCV

- Complexidade Ciclomática(McCabe - 1976)
  - Indica a complexidade de um programa
  - Mede a quantidade de caminhos execução independentes no código

$$M = E - N + 2 \times P$$

- **M**: Complexidade Ciclomática
- **E**: Quantidade de Setas
- **N**: Quantidade de Nós
- **P**: Quantidade de Componentes Conectados



NOA

- Conta diretamente o **número de atributos** de uma classe
- Pode identificar os potenciais problemas:
  - Uma classe com muitos atributos, pode indicar **coesão coincidente** e pode requerer mais **decomposição** para melhorar a **complexidade do modelo**
  - Classes com **nenhum atributo** devem ser analisadas com précisão para identificar se ela não seria uma classe utilitária ao invés de um classe comum
- Limiares comuns de NOA estão entre os valores **2 e 5***

*objecteering.com

AFS - Access to File System

- Conta o **número de funções** no código que fazem o **acesso direto ao disco**
- Em sistemas grandes ou complexos, pode ser utilizada para aumentar a atenção durante as manutenções do código, já que erros em **algoritmos** que trabalham diretamente com o sistema de arquivo podem ser **desastrosos**.
- Ao ser usada em conjunto com **CCV** e **NOA**, quando estas **também estiverem altas**, aumenta-se ainda mais o nível de criticidade da manutenção do código.
- (... trecho de código ->)

AFS - Access to File System

```
Map<String, Double> peso = new HashMap<String, Double>();

peso.put("read",0.25);
peso.put("write",0.50);
peso.put("rename",0.10);
peso.put("delete",0.10);
peso.put("append",0.50);

Map<String, String> busca = new HashMap<String, String>();
busca.put("read","\\.read(.*)\\(");
busca.put("delete", "\\.delete(.*)\\(");
busca.put("write", "\\.write(.*)\\(");
busca.put("append", "\\.append(.*)\\(");
busca.put("rename", "\\.renameTo(.*)\\(");
```

```
for (String valor: busca.keySet()) {
    pattern = Pattern.compile(busca.get(valor));
    matcher = pattern.matcher(codigo);
    count=0;
    //total=0.0;
    while (matcher.find()) {
        count++;
    }
    total+=peso.get(valor);
    total=total*count;
    valorF = BigDecimal.valueOf(total).setScale(2, RoundingMode.DOWN);
    //System.out.println(valor+":"+valorF);
}
System.out.println("AFS:"+valorF);
```

Solução

- Solução usando o Plugin do Eclipse - AST
- Executa as análise das 3 métricas sobre uma classe e fornece as seguinte saída

```
-------------------- Metrics of Software --------------------
Cyclomatic Complexity:18.0
-------------------------------------------------------------
Number of Attributes:3
-------------------------------------------------------------
AFS:2.34
-------------------------------------------------------------
```

Obrigado….