



**UNIVERSIDADE  
FEDERAL DO CEARÁ**  
CAMPUS DE QUIXADÁ

**CURSO DE ENGENHARIA DE SOFTWARE**

**RELATÓRIO – TRABALHO FINAL QUALIDADE DE SOFTWARE**  
**Projeto TERMLT**

**Equipe:**

**Antônio Xavier de Sousa Neto**

**Professora:**

**Carla Ilane Moreira Bezerra**

**QUIXADÁ**

**Março, 2021**

## SUMÁRIO

|     |   |                                      |
|-----|---|--------------------------------------|
| 1   | DESCRIÇÃO DO PROJETO .....  | 2                                    |
| 2   | AVALIAÇÃO DO PROJETO .....  | 2                                    |
| 2.1 | Medição 1 – Antes de refatorar o projeto.....                           | 2                                    |
| 2.2 | Detecção dos Code Smells .....  | 4                                    |
| 2.3 | Medição 2 – Após Refatorar Code Smell X .....                           | 5                                    |
| 2.4 | Medição 3 – Após Refatorar Code Smell Y .....                           | <b>Erro! Indicador não definido.</b> |
| 2.5 | Medição Z – Após a refatoração de todos os code smells do projeto ..... | 6                                    |
| 3   | COMPARAÇÃO DOS RESULTADOS .....   | 7                                    |
|     | REFERÊNCIAS .....   | 8                                    |
|     | APÊNDICE A .....  | 8                                    |

## 1 DESCRIÇÃO DO PROJETO

O projeto selecionado é um projeto Java OO e Web de Código Aberto, é uma ferramenta de gerenciamento de terminologia compatível com SKOS baseada em tecnologias da Web Semântica. Ele permite gerenciar vocabulários consistindo de thesauri e ontologias. Ele também pode gerenciar documentos que usam termos dos vocabulários e analisar os documentos para encontrar ocorrências desses termos. O projeto pode ser encontrado abaixo.

Link do projeto: <https://github.com/kbss-cvut/termi>

Link do GitHub com os Arquivos:

Tabela 1 – Características do Projeto

| Projeto | LOC    | # de classes | # de releases |
|---------|--------|--------------|---------------|
| TermIt  | 27.733 | 403          | 5             |

## 2 AVALIAÇÃO DO PROJETO

### 2.1 Medição 1 – Antes de refatorar o projeto

Nessa Seção deve ser incluída a Tabela com a medição das métricas de coesão, acoplamento, complexidade, herança e tamanho, antes do projeto ser refatorado. Para isso será utilizada a ferramenta Understand. A Tabela 2 apresenta a descrição das métricas, faça uma tabela similar.

Tabela 2 – Medição dos atributos antes de refatorar o projeto.

| Sistema                 | Coesão       | Complexidade |                     |     |        | Herança |                    |        | Acoplamento | Tamanho |      |      |     |
|-------------------------|--------------|--------------|---------------------|-----|--------|---------|--------------------|--------|-------------|---------|------|------|-----|
|                         | LCOM2        | ACC          | SCC                 | EVG | MaxNet | DIT     | NOC                | IFANIN | CBO         | LOC     | CLOC | NIM  | CDL |
| S1 antes da refatoração | <b>12355</b> | 358          | 3725<br><b>4705</b> | 443 | 179    | 678     | 197<br><b>1338</b> | 513    | <b>1651</b> | 277173  | 7585 | 3253 | 403 |
| S1 após refat. CS FE    | <b>12377</b> | 337          | 3585<br><b>4530</b> | 433 | 175    | 652     | 197<br><b>1319</b> | 470    | <b>1638</b> |         |      |      |     |



Tabela 3 – Métricas dos atributos internos de qualidade (MCCABE, 1976; CHIDAMBER; KEMERER, 1994; LORENZ; KIDD, 1994; DESTEFANIS *et al.*, 2014)

| Atributos    | Métricas   | Descrição   |
|--------------|--|---|
| Coesão       | <i>Lack of Cohesion of Methods (LCOM2)</i><br>(CHIDAMBER; KEMERER, 1994) | Mede a coesão de uma classe.<br>Quanto maior o valor dessa métrica, menos coesiva é a classe.   |
|              | <i>Coupling Between Objects (CBO)</i><br>(CHIDAMBER; KEMERER, 1994)      | Número de classes que uma classe está acoplada<br>Quanto maior o valor dessa métrica, maior é o acoplamento de classes e métodos.   |
| Complexidade | <i>Average Cyclomatic Complexity (ACC)</i><br>(MCCABE, 1976)             | Média da complexidade ciclomática de todos os métodos.<br>Quanto maior o valor dessa métrica, mais complexa são a classes e métodos.  |
|              | <i>Sum Cyclomatic Complexity (SCC)</i><br>(MCCABE, 1976)                 | Somatório da complexidade ciclomática de todos os métodos.<br>Quanto maior o valor dessa métrica, mais complexos são as classes e métodos.  |
|              | <i>Nesting (MaxNest)</i><br>(LORENZ; KIDD, 1994)                         | Nível máximo de aninhamento de construções de controle.<br>Quanto maior o valor dessa métrica, maior é a complexidade de classes e métodos.   |
|              | <i>Essential Complexity (EVG)</i><br>(MCCABE, 1976)                      | Mede o grau na qual um módulo contém construtores não estruturados.<br>Quanto maior o valor dessa métrica mais complexas são as classes e métodos.  |
| Herança      | <i>Number Of Children (NOC)</i><br>(CHIDAMBER; KEMERER, 1994)            | Número de subclasses de uma classe.<br>Quanto maior o valor dessa métrica maior é o grau de herança de um sistema.  |
|              | <i>Depth of Inheritance Tree (DIT)</i><br>(CHIDAMBER; KEMERER, 1994)     | O número de níveis que uma subclasse herda de métodos e atributos de uma superclasse na árvore de herança.<br>Quanto maior o valor dessa métrica maior é o grau de herança de um sistema. |
|              | <i>Base Classes (IFANIN)</i><br>(DESTEFANIS <i>et al.</i> , 2014)        | Número imediato de classes base.<br>Quanto maior o valor dessa métrica, maior o grau de herança de um sistema.  |
| Tamanho      | <i>Lines of Code (LOC)</i><br>(LORENZ; KIDD, 1994)                       | Número de linhas de código, excluindo espaços e comentários.<br>Quanto maior o valor dessa métrica, maior é o tamanho do sistema.   |
|              | <i>Lines with Comments (CLOC)</i><br>(LORENZ; KIDD, 1994)                | Número de linhas com comentários.<br>Quanto maior o valor dessa métrica maior o tamanho do sistema.   |
|              | <i>Classes (CDL)</i><br>(LORENZ; KIDD, 1994)                             | Número de classes. Quanto maior o valor , maior o tamanho do sistema.   |
|              | <i>Instance Methods (NIM)</i><br>(LORENZ; KIDD, 1994)                    | Número de métodos de instância. Quanto maior o valor dessa métrica maior é o tamanho do sistema.  |

## 2.2 Detecção dos Code Smells

Foram detectadas 308 instâncias de Code Smells, dentre eles estão: Shotgun Sugery, Refuse Parent Bequest, Intensive Coupling, God Class, Feature Envy, Dispersed Coupling e Data Class. Como são muitas instâncias e God Class e Data Class tem apenas 1 detecção. Optamos por escolher pelos outros 5 Smells que foram detectados, abaixo serão escolhidos 40 instâncias dessas 308.

Tabela 3 – Code smells do projeto.

| Nome do Code Smell | Quantidade |
|--------------------|------------|
|--------------------|------------|

|                               |    |
|-------------------------------|----|
| <b>Refused Parent Bequest</b> | 10 |
| <b>Feature Envy</b>           | 10 |
| <b>Dispersed Coupling</b>     | 8  |
| <b>Shotgun Sugery</b>         | 10 |
| <b>Intensive Coupling</b>     | 2  |

### 2.3 Medição 2 – Após Refatorar Code Smell: Feature Envy e Refused Parent Bequest

Os Code Smells Feature Envy e Refused Parent Bequest, foram removidos do projeto, onde uma porcentagem de cada um foi removida, devido terem muitos deles, Foram Removidos 10 FE e 10 RPB, ficando assim 298 após a refatoração. Para a retirada do FE, foi utilizado a técnica de fatoração de Move Method e Extract Method, onde o Move Method foi utilizado em 8 dos 10 FE, e o Extract em 2.

Logo após para a retirada do RPB, foi utilizado as técnicas de Replace Inheritance with Delegation, onde foi retirado as heranças que não era condizentes com as classes e também Extract Super class, onde foram criadas novas classes e delegado os métodos que faziam mais

sentido para elas e depois declarado a herança certa. Onde Replace Inheritance foi usada mais vezes que a Extract super class. Depois da fatora  o foi medida a qualidade interna, onde pode se notar em compara  o a antes da fatora  o uma melhora significativa nos atributos de heran  a e complexidade, onde antes Complexidade era 4695 e agora   4448 e Heran  a antes da fatora  o era 1388 e agora   1307. Tivemos tamb  m um pouco de melhora em Coes  o, e tamb  m em acoplamento.

Acredita-se que grande melhora e acoplamento se d   pelos m  todos de fatora  o e smells refatorados, onde foi movido muitos m  todos pra onde eles fariam mais sentidos e removido e criado heran  as tamb  m que faziam mais sentido.

#### **2.4 Medica  o 3 – Ap  s Refatorar Code Smell: Intensive Coupling**

Na refatora  o de Intensive Coupling. Como   um code smell que n  o existem m  todos oficiais de refatora  o, por  m   um code smell que utiliza os m  todos de sua classe excessivamente opera  es de outra, foi usada a estrat  gia de refatora  o de Move method, onde foi criado um m  todo dentro da classe onde chamava excessivamente esses m  todos de outra classe.

#### **2.5 Medica  o 3 – Ap  s Refatorar Code Smell: Dispersed Coupling**

O Dispersed Coupling tamb  m   um Code smell que n  o existem m  todos oficiais de refatora  o, ent  o , como o Intensive , o Dispersed tem m  todos da sua classe que acessam e dependem de muitas opera  es excessivas de outras classes, tamb  m foi usado o Move Method e Extract para criar um m  todo na classe sem precisar chamar de outras ou Mover para a classe certa.

#### **2.6 Medica  o 4 – Ap  s Refatorar Code Smell: Shotgun Surgery**

No caso do Shotgun, foi utilizado as t  cnicas de refatora  o de Move Field e Move method, e alguns casos, alguns getters e setters que estavam na classes e n  o era utilizados foram removidos, assim removendo esses m  todos que n  o eram utilizados.

## 2.7 Medição 5 – Após Refatorar Todos os Code Smells

No final, ao refatorar os Code Smells, abaixo na tabela mostrará o comparativo de como o código estava sem refatoração e depois para cada Code Smell.

| <b>Coesão</b> | <b>Complexidade</b> | <b>Herança</b> | <b>Acoplamento</b> |
|---------------|---------------------|----------------|--------------------|
| <b>12355</b>  | <b>4705</b>         | <b>1338</b>    | <b>1651</b>        |
| <b>12377↑</b> | <b>4530↓</b>        | <b>1319↓</b>   | <b>1638↓</b>       |
| <b>12390↑</b> | <b>4448↓</b>        | <b>1307↓</b>   | <b>1652↑</b>       |
| <b>12476↓</b> | <b>4572↑</b>        | <b>1330↑</b>   | <b>1676↑</b>       |
| <b>11809↓</b> | <b>4520↓</b>        | <b>1351↑</b>   | <b>1684↑</b>       |
| <b>12565↑</b> | <b>4530↑</b>        | <b>1319↓</b>   | <b>1638↓</b>       |

Na tabela está o comparativo entre antes da fatoração e depois, e o que melhorou ou piorou dentro do código, o aumenta da coesão no código(seta pra cima), indica um aumento positivo, pois o código está mais coeso após as fatorações, como pode ver na Coluna 1 e 2, após isso a Coesão decai um pouco, e por ultimo sobre novamente, se for compararmos na tabela noacima, a piora da coesão se dá pelo fato após as refatorações dos Smells Dispersed e Intensive Coupling, pelo fato de não haver muita experiência na suas refatorações. A mesma coisa acontece com a Complexidade, ela diminui de forma positiva e depois aumenta na parte da Refatoração do Dispersed e Intensive Coupling, e o mesmo acontece com Herança e por fim acoplamento que no final diminui depois da refatorações de Shotgun Surgery.

## 3 COMPARAÇÃO DOS RESULTADOS

Leia

o

artigo:

[https://www.sciencedirect.com/science/article/pii/S0950584920301142?casa\\_token=xcwL1BwaRFUAAAAA:wZjXB0Wx-0FiMSpZSzyi0b7iRe7ZJOr8FdwiHzEkvzeQHh0Iz6mxPCF769JgRiZ69TyfI5l8BP0](https://www.sciencedirect.com/science/article/pii/S0950584920301142?casa_token=xcwL1BwaRFUAAAAA:wZjXB0Wx-0FiMSpZSzyi0b7iRe7ZJOr8FdwiHzEkvzeQHh0Iz6mxPCF769JgRiZ69TyfI5l8BP0)

Faça uma comparação dos resultados do seu projeto de acordo com esse artigo.



## **REFERÊNCIAS**

AZEEM, Muhammad. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. *Information and Software Technology*, v. 108, p. 115-138, 2019.

SABIR, Fatima. A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems. *Software: Practice and Experience*, v. 49, n. 1, p. 3-39, 2019.

## **APÊNDICE A**

Incluir possíveis documentos que possam ser gerados no desenvolvimento do sistema.