# 3. METODOLOGIA

Este capítulo apresenta a ferramenta proposta. Uma visão geral da ferramenta é apresentada na Seção 3.1. A Seção 3.2 explica como é feito a configuração da ferramenta. Na Seção 3.3 é apresentado o ambiente em que a ferramenta foi inserida para comparação do resultado da análise de similaridade com outras ferramentas. Por fim, a Seção 3.4 apresenta como foi o experimento para obter os resultados dessas comparações.

# 3.1 A ferramenta

O objetivo principal deste trabalho é criar uma ferramenta adaptável baseada em contagem de atributos, que extrai as características-chave únicas da linguagem e o processamento dessas para criação de uma métrica que define o nível de similaridade entre dois códigos-fontes.

Essa ferramenta foi desenvolvida em linguagem C (o código-fonte pode ser encontrado no apêndice A) e permite uma configuração personalizada, pois qualquer linguagem de programação pode ser analisada, assim como as métricas de complexidade. No total foram implementadas 23 métricas de complexidade, que são baseadas nas técnicas utilizadas por Merlo e Hasltead (detalhes sobre essas técnicas podem serem vistos na Seção 2.6), em suas respectivas soluções. As métricas são listadas na Seção 3.2.

A ferramenta recebe quatro parâmetros. Os dois primeiros são os arquivos de configuração, sendo o primeiro para a linguagem e o segundo para as métricas de complexidade, e os outros dois os arquivos contendo os códigos-fontes a serem analisados.

Na Figura 3.1 é mostrado um exemplo de chamada da ferramenta. Neste exemplo, “dados.txt” é o arquivo contendo as características da linguagem a ser analisada e “métricas.txt” é o arquivo contendo as métricas de complexidade que serão calculadas para definir o nível de similaridade entre “Exemplo1-1.c” e “Exemplo1-2.c”, que são os dois códigos-fontes comparados. Os dois primeiros arquivos são detalhados na Seção 3.2. Na Figura 3.2 é mostrado a saída deste exemplo.

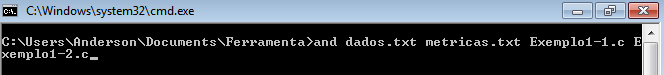


Figura 3.1 - Exemplo da chamada da ferramenta

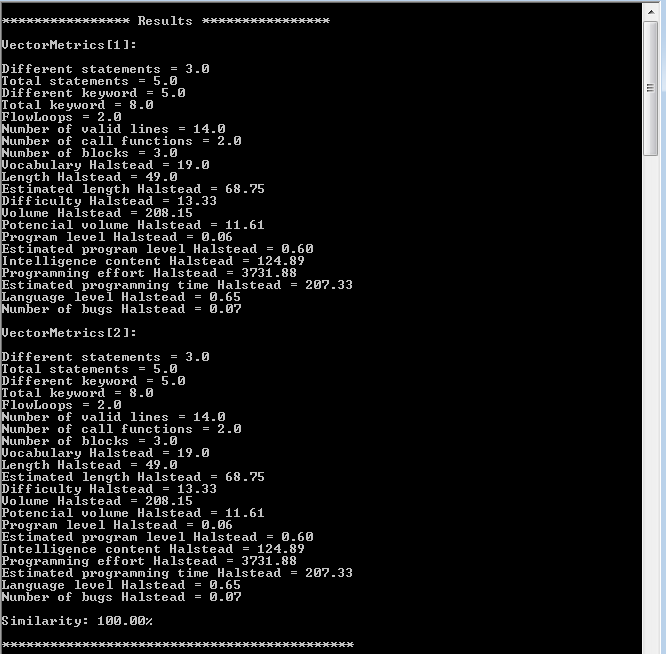


Figura 3.2 - Resultado da execução da ferramenta

# 3.2 Configurando a ferramenta

Como visto na Seção 3.1, a ferramenta é adaptável para qualquer linguagem de programação. No exemplo da Figura 3.1, o arquivo “dados.txt” é o arquivo que contém as características da linguagem dos códigos que serão comparados, ou seja, qualquer linguagem de programação pode ser analisada por essa ferramenta, apenas passando as informações que são necessárias para a comparação. Na Figura 3.3 é apresentado o arquivo “dados.txt”.

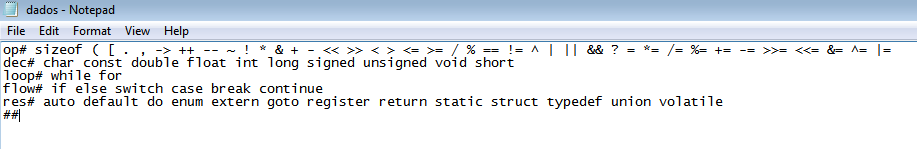


Figura 3.3 - Arquivo de configuração da linguagem

No exemplo da Figura 3.1, a linguagem dos arquivos analisados é C, assim o arquivo “dados.txt” é composto por cinco linhas que configuram a ferramenta, sendo que cada linha possui características específicas da linguagem.

A ferramenta reconhece cada característica por causa da identificação da linha, que vai desde o primeiro caractere até o caractere ‘#’. Por exemplo, na primeira linha do arquivo “dados.txt” o identificador é “op”, assim nessa linha são configurados os operadores da linguagem que serão avaliados. Cada linha do arquivo que configura a linguagem da ferramenta é composta por “identificador + # + conteúdo separado por espaço em branco” e a ultima linha contém a *string* “##” para determinar o fim do arquivo.

No exemplo da Figura 3.1, o arquivo “métricas.txt” é o arquivo que contém as métricas de complexidade que serão calculadas para obter a similaridade entre os códigos comparados. Na Figura 3.4 é apresentado o arquivo “metricas.txt”. Cada linha do arquivo configura uma métrica.

A ferramenta reconhece uma métrica por causa da identificação da linha, que vai desde o primeiro caractere até o caractere ‘#’. Por exemplo, na primeira linha do arquivo “metricas.txt” o identificador é “opDiff”, assim é configurado a métrica que calcula os operadores distintos encontrados no código. Abaixo são apresentadas as métricas com seus respectivos identificadores:

1. Número de operadores distintos → opDiff
2. Número total de operadores → opTotal
3. Número de *keywords* de declaração distintos → decDiff
4. Número total de *keywords* de declaração → decTotal
5. Número de *keywords* distintos → resDiff
6. Número total de *keywords* → resTotal
7. Número total estimado de estruturas semânticas → flowLoop
8. Número de linhas → numberLines
9. Número de chamadas de funções → numberCallFunctions
10. Número de blocos de códigos → numberBlocks
11. Vocabulário → vocabHalstead
12. Tamanho → lengthHalstead
13. Tamanho estimado → estimatedLengthHalstead
14. Volume → volumeHalstead
15. Volume potencial → potencialVolumeHalstead
16. Nível → programLevelHalstead
17. Nível estimado → estimatedProgramLevelHalstead
18. Dificuldade → diffHalstead
19. Conteúdo inteligente → intelligenceContentHalstead
20. Esforço de programação → programmingEffortHalstead
21. Tempo estimado de programação → estimatedProgrammingTimeHalstead
22. Nível da linguagem → languageLevelHalstead
23. Número de *bug* → numberBugsHalstead

Cada linha de configuração, com exceção da última que contém a *string* “##” para determinar o fim do arquivo, é composta de vários parâmetros separados por espaço em branco. O primeiro parâmetro serve para indicar para a ferramenta que a métrica está habilitada, se o valor for 1 está habilitada ou se o valor for 0 está desabilitada. O segundo parâmetro representa o limiar de comparação, pode ser qualquer valor entre 0 e 1. O terceiro parâmetro representa o peso atribuído a métrica, pode ser qualquer valor positivo e diferente de 0. Na Figura 3.4 é possível verificar que as duas primeiras métricas (operadores distintos e total de operadores) estão desabilitadas, por isso o resultado é como o mostrado na Figura 3.2.

Caso queira implementar uma nova métrica, é necessário modificar o arquivo que faz a configuração e implementar o cálculo da métrica no código da ferramenta (ver apêndice A).

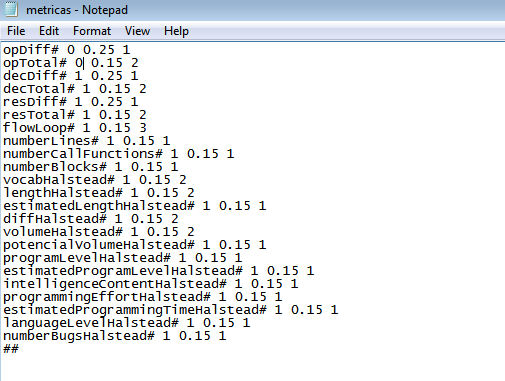


Figura 3.4 – Arquivo de configuração das métricas de complexidade

# 3.3 O Ambiente de comparação

Para comparação do resultado da análise de similaridade com outras ferramentas, a ferramenta proposta foi inserida em um ambiente de testes desenvolvido por Danilo (2014) chamado Sistema de Análise de Similaridade, que possui algumas funcionalidades detalhadas a seguir.

**Selecionar arquivos**

A Figura 3.5 retirada de Danilo (2014) mostra a página para submissão de arquivos para análise de similaridade, por meio da qual os arquivos são selecionados de um diretório e submetidos para comparação.

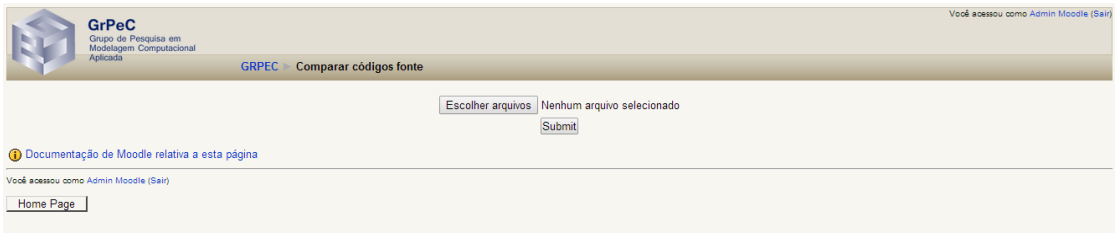


Figura 3.5 – Submissão externa - Página de submissão

**Selecionar cada ferramenta e a técnica de normalização**

Na Figura 3.6 retirada de Danilo (2014) mostra a página para selecionar cada ferramenta, assim como a técnica de normalização.

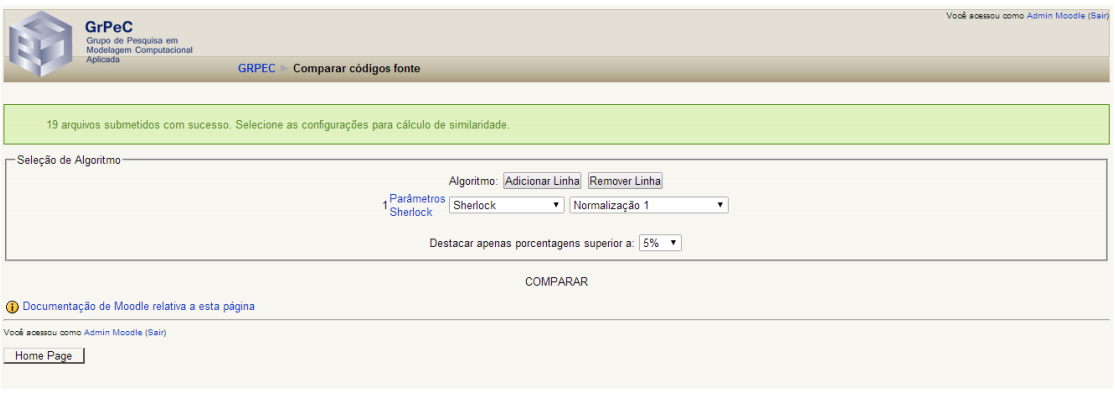


Figura 3.6 – Submissão externa - Página de comparação

**Execução da comparação**

Após a configuração dos algoritmos e das normalizações, pode-se executar a comparação dos códigos-fonte submetidos. Na Figura 3.7 retirada de Danilo (2014) é apresentado o resultado da comparação do Sherlock com duas técnicas de normalização. Todos os códigos são combinados 2 a 2, apresentando a similaridade para cada par de código em uma célula. De acordo com Danilo (2014), os pares com similaridade superior a 95% são destacados com a cor vermelha, os pares com similaridade igual a 0% são marcados com a cor verde e todos os demais com a cor laranja.

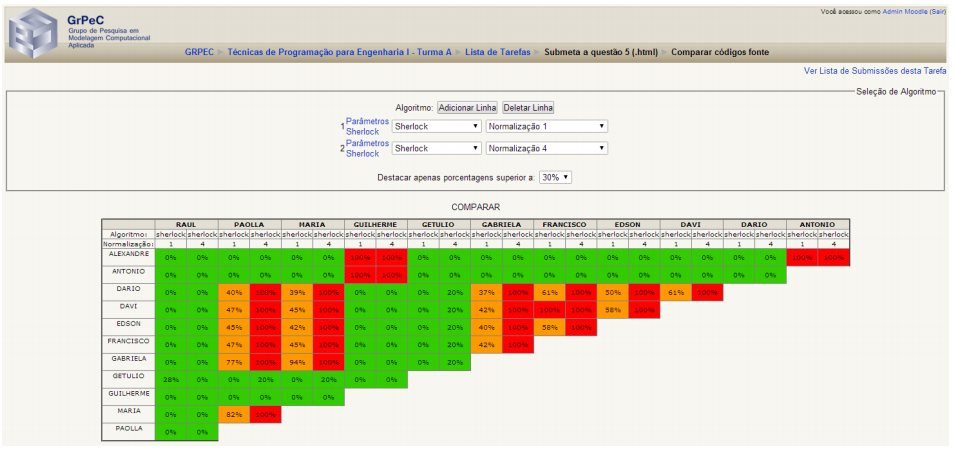


Figura 3.7 - Interface com execução de um algoritmo

# 3.4 O Experimento