Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Barbara Malta Gomes
Orientadora: Profa. Dra. Mariza Andrade da Silva Bigonha
Coorientadora: Profa. Dra. Kecia Aline Marques Ferreira

Relatótrio Técnico-Científico de Inicição Científica

Belo Horizonte - MG
$2010/2^o$ Semestre

Sumário

1 Introdução **3**

2 Coleta Automática na Ferramenta CONNECTA **3**
2.1 Implementação 4
2.1.1 Algoritmo para Coleta Automática 4
2.2 Interfaces com o Usuário 4
2.3 Conclusão 5

3 Recálculo *KB3* **5**
3.1 Implementação 6
3.2 Interfaces com o Usuário 6
3.3 Conclusão 6

4 Simulador - Ferramenta para Simulação de Alterações em Código de Software **7**
4.1 Tipo de Modificação Simulada 7
4.2 Especificação de Requisitos 8
4.2.1 Casos de Uso 8
4.2.2 Interfaces com o Usuário 8
4.3 Arquitetura 15
4.3.1 Implementação 16
4.3.2 Algoritmo para Simulação de Modificação em Métodos 16
4.4 Conclusão 20

5 Conclusão **20**

6 Referências Bibliográficas **20**

1 Introdução

O presente relatório apresenta a descrição das atividades realizadas pela autora, aluna de graduação do curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), em seus trabalhos de Iniciação Científica no período de agosto de 2010 a fevereiro de 2011, sob orientação da professora Dra. Mariza Andrade da Silva Bigonha e coorientação da professora Dra. Kecia Aline Marques Ferreira.

O trabalho realizado consistiu em três etapas, sendo elas:

1. **Implementação da coleta automática na ferramenta CONNECTA.** Implementação do módulo de coleta automática de dados na ferramenta Connecta, desenvolvida por Ferreira [1] em seu mestrado com a finalidade de avaliar softwares em Java. CONNECTA analisa arquivos de *bytecodes*, coletando métricas para avaliação da coesão, estabilidade e conectividade dos programas. Esse novo módulo permite que sejam fornecidos ao CONNECTA vários programas para analise, sendo cada um, em pastas distintas.

2. **Implementação do recálculo da métrica *KB3*:** A métrica *KB3* foi desenvolvida Ferreira [2] em seu doutorado, com a finalidade de avaliar o esforço para realizar alterações em *softwares*. O recálculo da métrica possibilita a modificação das constantes usadas em seu cálculo, com a finalidade de "calibrá-las".

3. **Implementação da ferramenta de simulação de modificação em *software*.** A ferramenta de simulação foi implementada para avaliar em quantos passos (quantas classes são visitadas) quando se faz alterações nos parâmetros ou no nome de um método, fornecendo um valor médio de esforço para alterações no *software*. Seu objetivo principal é verificar se a métrica *KB3* fornece valores consistentes.

O presente relatório visa apresentar, em cada uma das seções que se seguem, a descrição de forma detalhada de cada etapa realizada durante o trabalho, explicando os conceitos, assim como a metodologia utilizada.

Seção 2 descreve em detalhes as modificações efetuadas na ferramenta CONNECTA, para que seja possível realizar a coleta automática de dados. Esta facilidade propicia ao usuário da ferramenta fornecer mais de um programa por vez, separados por pastas, e, nesse caso, a ferramenta considera cada pasta no diretório fornecido como sendo um programa. A partir daí realiza a coleta dos dados e os salva em arquivo no mesmo diretório do programa em questão.

Seção 3 apresenta os passos para a implementação do recálculo da métrica *KB3*, realizado por meio da inserção de novos valores para as constantes utilizadas na fórmula K3B.

Seção 4 descreve sobre a implemetação da nova ferramenta de simulação de modificação de alteração nos métodos de um *software*, construída utilizando a linguagem Java. Aborda também como utilizá-lo e os conceitos aplicados.

Seção 5 conclui esse relatório.

2 Coleta Automática na Ferramenta CONNECTA

A primeira versão da ferramenta CONNECTA fornecia a opção para coleta de dados de apenas um *software* por vez, e dessa maneira, caso o usuário tivesse mais de um *software* a fim de realizar a análise fornecida pelo CONNECTA, era necessário fornecer um *software* por vez à CONNECTA, ficando dependente durante toda a coleta dos dados.

A implementação da coleta automática possibilita ao usuário informar um diretório contendo apenas pastas. Cada pasta presente nesse diretório é considerada um *software* distinto, e dessa

forma, a ferramenta realiza a coleta dos dados de forma separada para cada pasta e armazena os resultados em arquivo, que por sua vez são salvos na pasta de cada *software*.

2.1 Implementação

Utilizando a IDE *NetBeans*, a mesma utilizada na implementação da ferramenta CONNECTA, assim como a linguagem Java, realizou-se a implementação da modificação desejada.

Criou-se na tela principal do CONNECTA um item de menu, que permite ao usuário acessar o módulo de coleta automática da ferramenta. Inseriu-se, dentre as classes do CONNECTA, um novo formulário *JFrame* no qual é possível selecionar os diretórios que serão usados pela coleta automática. Nele esta contido também a lógica usada para arquivar os dados assim como a realização da coleta de cada *software* separadamente.

Um outro formulário *JFrame*, também foi criado a fim de informar ao usuário em qual nível a coleta está, ou seja, qual *software* esta sendo analisado no momento.

2.1.1 Algoritmo para Coleta Automática

```
1  para cada pasta pertencente ao diretorio selecionado faca
2
3          obtenha o caminho da pasta
4
5          altere o nome da pasta sendo analisada na interface com o usuario
6
7          obtenha os arquivos .class presentes na pasta
8
9          analise os arquivos .class
10
11 fim para
```

2.2 Interfaces com o Usuário

Para utilizar a coleta automática, é necessário acessar as seguintes interfaces com o usuário: *Tela Inicial* e *Seleção de diretório para coleta automática*. Durante a coleta automática dos dados o usuário tem contato com a interface *Pasta sendo analisada*, para que o mesmo esteja ciente em que passo está a coleta. A seguir são descritas as telas de acordo com o que precisa ser acessado para obter a coleta automática:

- *Tela Inicial.* Tela de apresentação da ferramenta. A Figura 1 apresenta sua interface. Para acessar a coleta automática, basta selecionar no menu *Arquivo*, a opção *Nova analise automática*

Figura 1: Interface de usuário *Tela Inicial*

- *Seleção de diretório para coleta automática.* Tela para seleção do diretório que contém as pastas para análise automática. A Figura 2 apresenta sua interface e a Tabela 1 descreve os comandos dessa interface e suas respectivas ações.

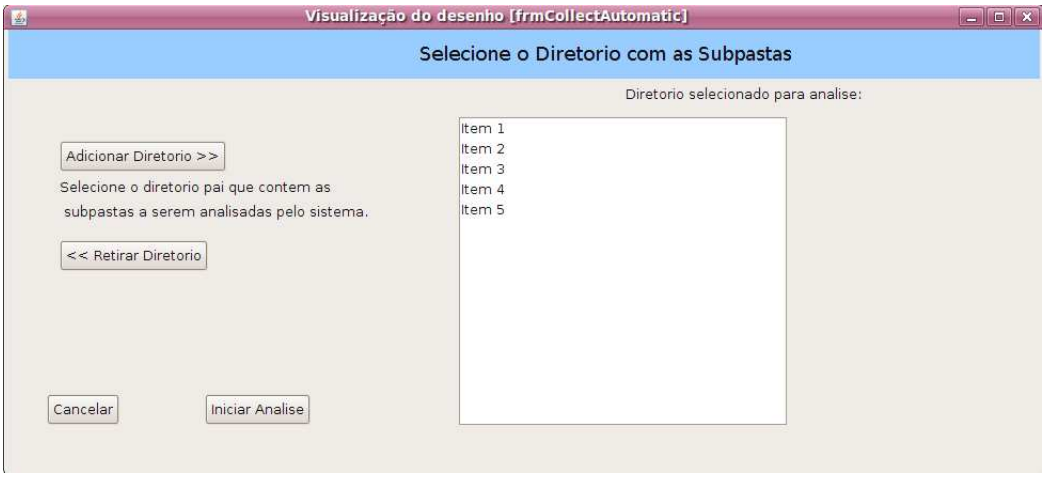

Figura 2: Interface de usuário *Seleção de diretório para coleta automática*

Comando	*Ação*
Adicionar diretório	Exibe uma caixa de diálogo que permite a seleção do diretório para coleta
Retirar diretório	Remove o diretório selecionado da lista de diretórios para coleta
Iniciar Análise	Inicia a coleta dos dados para cada pasta nos diretórios
Cancelar	Cancela coleta automática e retorna a *Tela Inicial*

Tabela 1: Comandos da interface de usuário *Seleção de diretório para coleta automática.*

- *Pasta sendo analisada.* Tela que exibe a pasta sendo analisada pelo CONNECTA. A Figura 3 apresenta sua interface.

Figura 3: Interface de usuário *Pasta sendo analisada*

2.3 Conclusão

Nessa seção foi descrita a alteração introduzida na ferramenta CONNECTA, afim de possibilitar ao usuário selecionar mais de um *software* por vez para coleta de dados pela ferramenta.

3 Recálculo *KB3*

O cálculo da métrica *KB3* inclui o uso de 4 constantes de acoplamento que influênciam no cálculo do valor dessa métrica. Os atributos são:

1. *Conexão por atributo.* Peso para as conexões entre as classes realizadas por meio de atributos.

2. *Conexão por valor.* Peso para as conexões entre as classes realizadas por meio de valores.

3. *Conexão por referência.* Peso para as conexões entre as classes realizadas por meio de referências.

4. *Conexão por herança.* Peso para as conexões entre as classes realizadas por meio de heranças.

Para que fosse possível obter os valores ideais para essas constantes, de modo que a métrica obtenha os resultados esperados, implementou-se no CONNECTA uma tela que permite, logo após ter sido feita a coleta de dados, alterar o valor usado para o cálculo da métrica.

3.1 Implementação

A implementação desse módulo foi feita utilizando a IDE *NetBeans* e a linguagem Java, sendo apenas incorporado ao código de CONNECTA uma classe *JFrame* que permite a inserção dos novos valores das constantes, assim como a adição de um botão na tela de exibição de histórico que redireciona para o *JFrame* citado anteriormente.

3.2 Interfaces com o Usuário

Para acessar a alteração dos valores das contantes na métrica *KB3*, basta abrir o resultado salvo da coleta do *software* desejado e logo após clicar no botão *Alterar Pesos de Acoplamento*, como mostra a Figura 4.

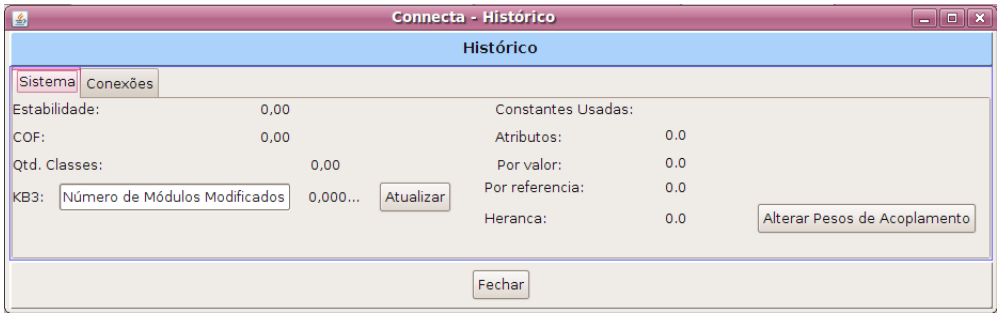

Figura 4: Interface de usuário *Show History*

Feito isso, é exibida a interface *Alterar pesos de acoplamento*, como mostra a Figura 5, onde devem ser inseridos os valores correspondentes a cada novo peso. Após a definição dos novos valores, basta clicar no botão *Recalcular* que a métrica é recalculada utilizando os novos valores das constantes.

3.3 Conclusão

Nessa seção foi descrita a alteração na ferramenta CONNECTA para que seja possível, após ter sido realizada a coleta pela ferramenta, recalcular o valor da métrica *KB3* mediante a inserção de novos valores para as constantes de acoplamento utilizadas na mesma.

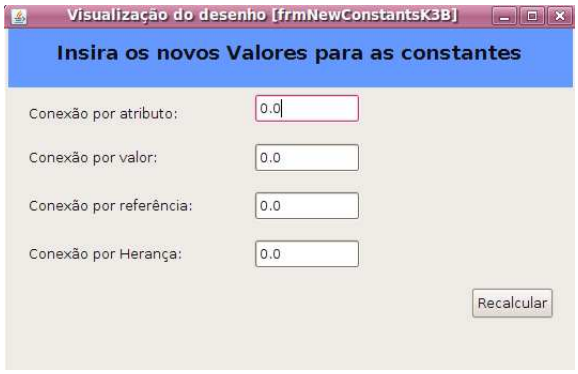

Figura 5: Interface de usuário *Alterar pesos de acoplamento*

4 Simulador - Ferramenta para Simulação de Alterações em Código de Software

Para ser identificado com precisão, quantas e quais são as classes que deverão ser visitadas para realizar uma alteração em qualquer parte do código de um *software*, é necessário a realização de uma análise passo a passo do mesmo, em nível de código.

Em *softwares* pequenos, onde é possível se ter o controle completo do seu código, sabendo o caminho para cada alteração a ser feita, estimular, ou até mesmo contar quantos passos são necessários para realizar tal alteração, pode ser, ou parecer, uma tarefa simples. Porém, em *softwares* grandes, no qual, por exemplo localizar as ocorrências de um determinado método, é uma tarefa mais complexa que requer tempo e concentração do programador. A única solução a ser utilizada é um outro *software* que simula a modificação em certa parte do código.

O *Simulador* analisa os executáveis de *softwares*, construídos utilizando a liguagem Java, afim de encontrar os pontos de conexões ocorrentes. Dessa maneira, pode-se contar quantos passos são necessários para realizar uma determinada modificação, assim como armazenar o caminho de tal alteração.

4.1 Tipo de Modificação Simulada

Existem várias alterações que podem ser realizadas em um *software*, podemos ver as mais recorrentes na Tabela 2. Na implementação atual é simulada apenas a alteração em métodos, porém, o *Simulador* permite a inclusão de outros tipos de modificação.

Remover Classe
Remover Interface
Remover Método
Alterar Valor de Referência
Renomear Método
Alterar parâmetros
Adicionar Herança
Remover Herança
Adicionar Atributo

Tabela 2: Exemplos de tipos de modificações em Software

4.2 Especificação de Requisitos

A especificação de requisitos do *Simulador* é constituída pelo objetivo da ferramenta, pelo diagrama de casos de uso e pela descrição das interfaces de usuário.

O *Simulador* tem por objetivo simular modificações em um *Software* de maneira a obter todos os passos necessários para a realização dessas, bem como a geração e a consulta de histórico dos resultados das simulações.

4.2.1 Casos de Uso

A Tabela 3 lista os casos de uso principais da ferramenta e seus respectivos objetivos. Considerando o ator como sendo *Usuário*, podemos ver na Figura 6 o digrama de Casos de Uso.

Casos de Uso	Descrição
Consultar Histórico	Consultar dados de simulações realizadas
Simular Software	Simulação de modificações em um Software

Tabela 3: Lista de Casos de Uso do *Simulador*

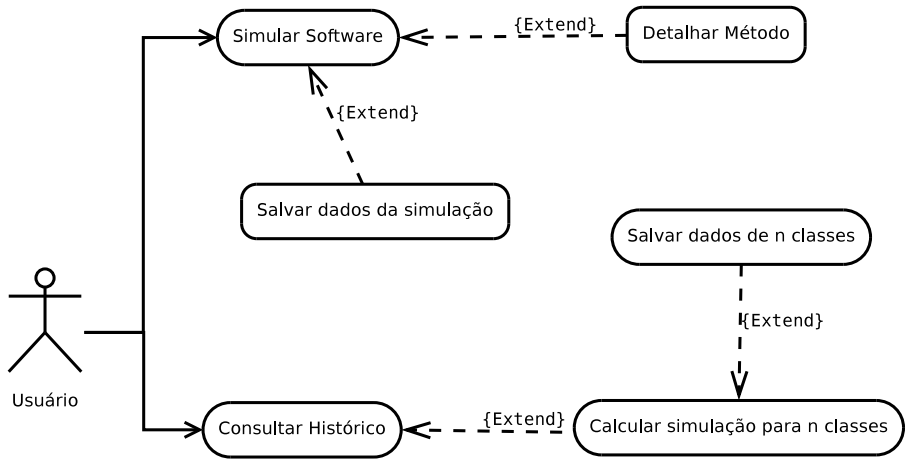

Figura 6: Diagrama de Casos de Uso do *Simulador*

4.2.2 Interfaces com o Usuário

O *Simulador* possui as seguintes interfaces com o usuário:

- Tela Inicial,
- Seleção de Classes para Simulação,
- Escolha da Modificação,
- Modificação em Métodos,
- Resultado da Simulação em Métodos,
- Detalhar Simulação de Método,
- Calcular Simulação em Métodos para N Classes,

- Coleta Automática de Dados,
- Seleção de Arquivo para Consulta de Dados de Simulação,
- Exibir Resultado da Consulta.

A seguir é apresentada a descrição dessas interfaces em detalhes.

- *Tela Inicial.* É a tela de apresentação da aplicação. A Figura 7 apresenta sua interface, enquanto a Tabela 4 descreve os comandos dessa interface e suas respectivas ações.

Comando	*Ação*
Simulação	Exibe a interface de usuário *Seleção de Classes para Simulação.*
Simulação Automática	Exibe interface de usuário *Coleta Automática de Dados.*
Consultar em arquivo	Exibe interface de usuário *Seleção de Arquivo para Consulta de Dados de Simulação.*
Sair	Finaliza a aplicação

Tabela 4: Comandos da interface de usuário *Tela Inicial*

Figura 7: Interface de usuário *Tela Inicial*

- *Seleção de Classes para Simulação.* Interface para o usuário selecionar as classes a serem utilizadas para simulação. A Figura 8 apresenta sua interface, enquanto a Tabela 5 descreve os comandos dessa interface e suas respectivas ações e a Tabela 6 descreve os campos da mesma.

Comando	*Ação*
Adicionar Arquivos	Exibe caixa de diálogo para que o usuário selecione os arquivos .class e os insira na lista de arquivos para simulação.
Excluir Arquivos	Retira os arquivos .class selecionados da lista para simulação.
Simular	Exibe interface de usuário *Escolha da Modificação.*
Voltar	Retorna a interface *Tela Inicial*

Tabela 5: Comandos da interface de usuário *Seleção de Classes para Simulação.*

Campo	*Descrição*	*Tipo*
Classes selecionadas para simulação	Classes adicionadas para simulação	Lista de Seleção Múltipla

Tabela 6: Campos da interface de usuário *Seleção de Classes para Simulação*.

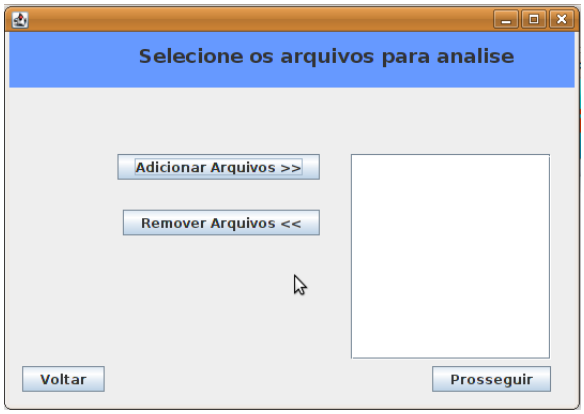

Figura 8: Interface de usuário *Seleção de Classes para Simulação*

- *Escolha da Modificação.* Oferece ao usuário os tipos de modificações disponíveis para simulação, sendo permitida apenas a seleção de uma modificação por vez. A Figura 9 apresenta sua interface, enquanto a Tabela 7 descreve os comandos dessa interface e suas respectivas ações e a Tabela 8 descreve os campos da mesma.

Comando	*Ação*
Prosseguir	Analisa as classes selecionadas e exibe a interface de usuário correspondente ao tipo de modificação selecionada
voltar	Retorna a interface de usuário *Seleção de Classes para Simulação*

Tabela 7: Comandos da interface de usuário *Escolha da Modificação*.

Campo	*Descrição*	*Tipo*
Modificação em Métodos	Seleciona a simulação de modificação em métodos	Botão de seleção em um grupo

Tabela 8: Campos da interface de usuário *Seleção de Escolha da Modificação*.

- *Modificação em Métodos.* Após a analise dos dados necessários para a simulação de modificação em métodos, essa interface é exibida afim de definir as opções de modificação em métodos. A Figura 10 apresenta sua interface, enquanto a Tabela 9 descreve os comandos dessa interface e suas respectivas ações, e, a Tabela 10 descreve os campos da mesma.
- *Resultado da Simulação em Métodos.* Interface para exibir os dados coletados durante a simulação de métodos. A partir dela é possível gravar os dados da simulação. A Figura 11 apresenta sua interface, enquanto a Tabela 11 descreve os comandos dessa interface e suas respectivas ações e a Tabela 12 descreve os campos da mesma.

Comando	*Ação*
Simular	Realiza a simulação dos métodos selecionados e exibe o resultado na interface de usuário *Resultado da Simulação em Métodos.*
Voltar	Retorna a interface de usuário *Escolha da Modificação*
Cancelar	Encerra o programa

Tabela 9: Comandos da interface de usuário *Modificação em Métodos.*

Campo	*Descrição*	*Tipo*
Lista de Parmetros	Opção de simular a modificação em lista de parâmetros do método	Botão de seleção em um grupo
Renomear método	Opção de simular a modificação de renomeação de método	Botão de seleção em um grupo
Todos os métodos do Software	Opção de simular todos os métodos contidos nos arquivos .class previamente selecionados	Botão de seleção em um grupo
Selecionar métodos	permite ao usuário escolher os métodos que serão simulados	Botão de seleção em grupo
Interfaces de parada	Exibe todas as classes do tipo Interface	Lista de seleção múltipla
Métodos Existentes	Exibe os métodos presentes nos arquivos .class	Lista de seleção múltipla

Tabela 10: Campos da interface de usuário *Modificação em Métodos.*

Comando	*Ação*
Detalhar	Exibe a interface de usuário *Detalhar Simulação de Método*
Salvar em Arquivo	Abre diálogo solicitando o caminho e o nome do arquivo para gravação e grava os resultados em tal arquivo
Voltar	Retorna a interface de usuário *Seleção de Classes para Simulação*
Cancelar	Encerra o programa

Tabela 11: Comandos da interface de usuário *Resultado da Simulação em Métodos.*

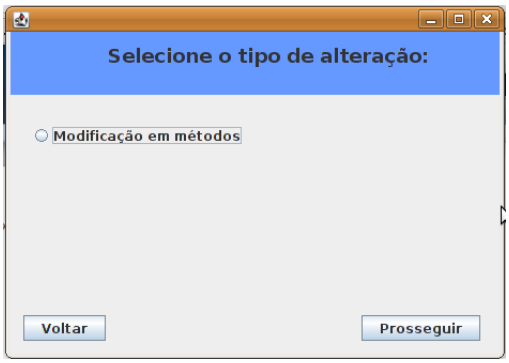

Figura 9: Interface de usuário *Seleção de Escolha da Modificação*

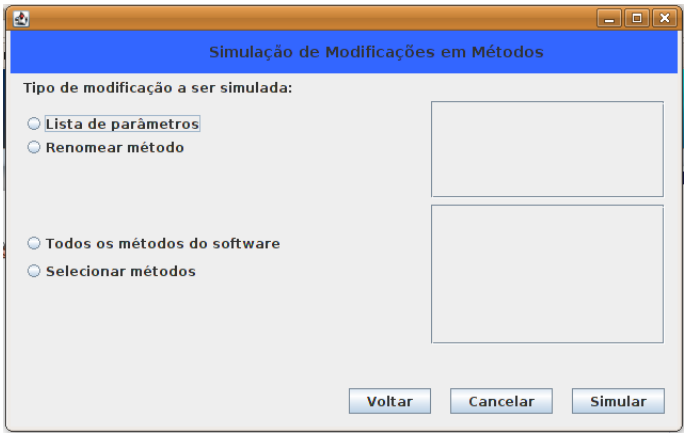

Figura 10: Interface de usuário *Modificação em Métodos*

Campo	*Descrição*	*Tipo*
Tipo de modificação simulada	Exibe o tipo de modificação escolhida	Texto não editável
Interfaces consideradas como ponto de parada	Exibi as interfaces de parada	Texto não editável
Classe	Nome da classe do sistema	Texto não editável
Método	Nome do método do sistema	Texto não editável
Número de passos	Número de passos para simulação do método do sistema	Texto não editável
Média do número de passos	Média do número de passos simulados	Texto não editável

Tabela 12: Campos da interface de usuário *Resultado da Simulação em Métodos*.

- *Detalhar Simulação de Método.* Exibe todos os detalhes, da simulação de modificação em métodos, do método selecionado previamente pelo usuário. Mostra o caminho a ser feito para a alteração de tal método. A Figura 12 apresenta sua interface, enquanto a Tabela 13 descreve os comandos dessa interface e suas respectivas ações e a Tabela 14 descreve os campos da mesma.

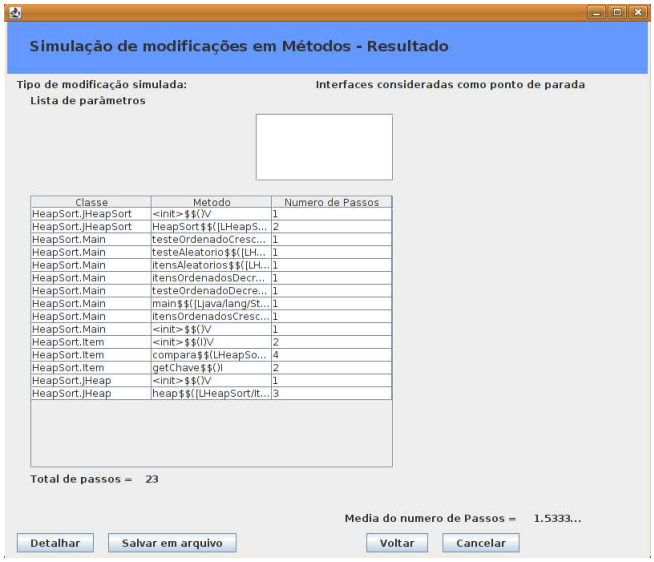

Figura 11: Interface de usuário *Resultado da Simulação em Métodos*

Comando	*Ação*
Terminar	Finaliza o programa
Voltar	Retorna a interface de usuário *Resultado da Simulação em Métodos*

Tabela 13: Comandos da interface de usuário *Detalhar Simulação de Método.*

Campo	*Descrição*	*Tipo*
Classe	Nome da classe do sistema	Texto não editável
Método	Nome do método do sistema	Texto não editável

Tabela 14: Campos da interface de usuário *Detalhar Simulação de Método.*

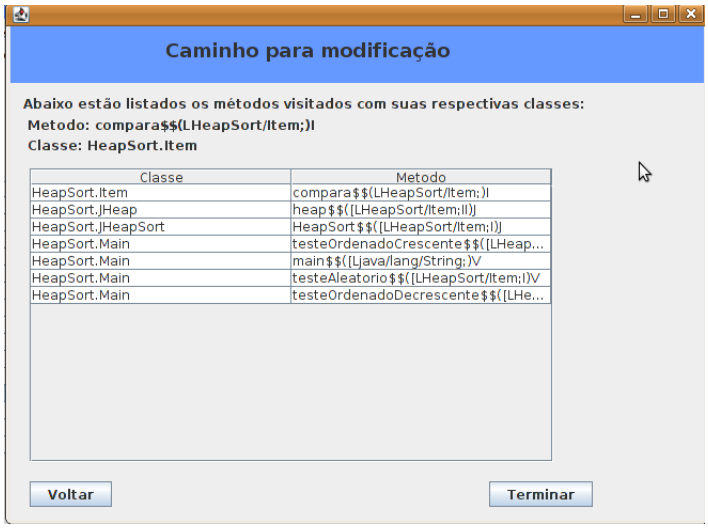

Figura 12: Interface de usuário *Detalhar Simulação de Método*

- *Calcular Simulação em Métodos para N Classes.* Permite ao usuário indicar quantas classes estariam em manutenção e então calcular a média de passos para a quantidade N determinadas de tais classes. A Figura 13 apresenta sua interface, podemos ver também na Figura 14 a exibição do resultado do cálculo, enquanto a Tabela 15 descreve os comandos dessa interface e suas respectivas ações e a Tabela 16 descreve os campos da mesma.

Comando	*Ação*
Calcular	Calcula a média de passos para n classes
Voltar	Retorna a interface de usuário *Exibir Resultado da Consulta*
Salvar	Salva o cálculo feito em um arquivo .txt

Tabela 15: Comandos da interface de usuário *Calcular Simulação em Métodos para N Classes.*

Campo	*Descrição*	*Tipo*
Valor de N	Valor escolhido para N	Texto Editável

Tabela 16: Campos da interface de usuário *Calcular Simulação em Métodos para N Classes.*

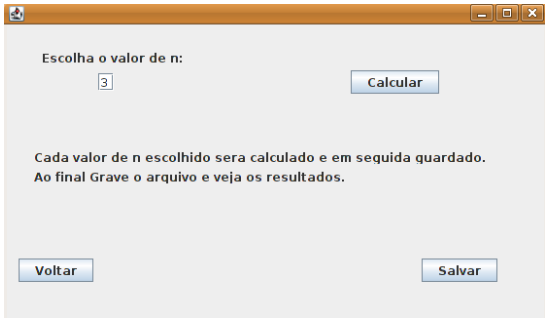

Figura 13: Interface de usuário *Calcular Simulação em Métodos para N Classes*

Figura 14: Interface de usuário que exibe o valor para N classes em manutenção

- *Coleta automática de dados.* A interface de seleção de arquivos *.class* para a coleta automática é a mesma utilizada para a simulação manual, porém, ao invés de selecionarmos os arquivos diretamente, selecionamos um diretório raiz, que contém os subdiretórios a serem analisados separadamente.

  Cada subdiretório contido na raiz é considerado como um conjunto de arquivos *.class* que serão analisados separadamente dos outros. A coleta automática está programada para selecionar a modificação em métodos, assim como os parâmetros *Modificação em Lista de Parâmetros* e *Selecionar todos os métodos* da mesma. Ao terminar, o resultado é gravado em arquivo no mesmo diretório da subpasta em questão.

- *Seleção de arquivos para Consulta de Dados de Simulação.* Interface para que o usuário selecione o arquivo que contém os dados de simulação que deseja consultar.A Figura 15 apresenta sua interface, enquanto a Tabela 17 descreve os comandos dessa interface e suas respectivas ações e a Tabela 18 descreve os campos da mesma.

- *Exibir Resultado da Consulta.* Exibe os dados armazenados em arquivo de simulações já realizadas. A Figura 16 apresenta sua interface, enquanto a Tabela 19 descreve os

Comando	*Ação*
Selecionar Arquivo	Exibe diálogo para seleção de arquivo que contém os dados
Mostrar Resultado	Exibe a interface com usuário *Exibir Resultado de Consulta*
Cancelar	Fecha a interface

Tabela 17: Comandos da interface de usuário *Seleção de arquivos para Consulta de Dados de Simulação.*

Campo	*Descrição*	*Tipo*
Arquivo	Nome do arquivo que contém os dados a serem consultados	Texto não Editável

Tabela 18: Campos da interface de usuário *Seleção de arquivos para Consulta de Dados de Simulação.*

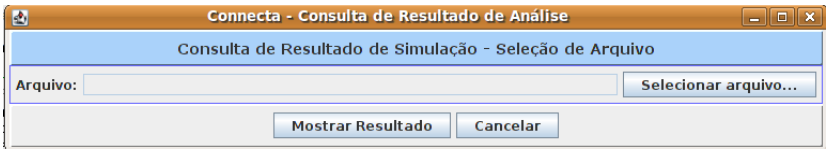

Figura 15: Interface de usuário *Seleção de arquivos para Consulta de Dados de Simulação*

comandos dessa interface e suas respectivas ações e a Tabela 20 descreve os campos da mesma.

Comando	*Ação*
Detalhar	Exibe a interface de usuário *Detalhar Simulação de Método*
Calcular média para N	Exibe a interface de usuário *Calcular Simulação em Métodos para N Classes*
Voltar	Fecha a interface

Tabela 19: Comandos da interface de usuário *Exibir Resultado da Consulta.*

Campo	*Descrição*	*Tipo*
Classe	Nome da classe do sistema	Texto não editável
Método	Nome do método do sistema	Texto não editável
Número de passos	Número de passos para simulação do método do sistema	Texto não editável
Média do número de passos	Média do número de passos simulados	Texto não editável

Tabela 20: Campos da interface de usuário *Exibir Resultado da Consulta.*

4.3 Arquitetura

A arquitetura do *software* agrupa as classes em pacotes lógicos:

- *Classes de Apresentação:* contém as classes responsáveis pela interação com o usuário.

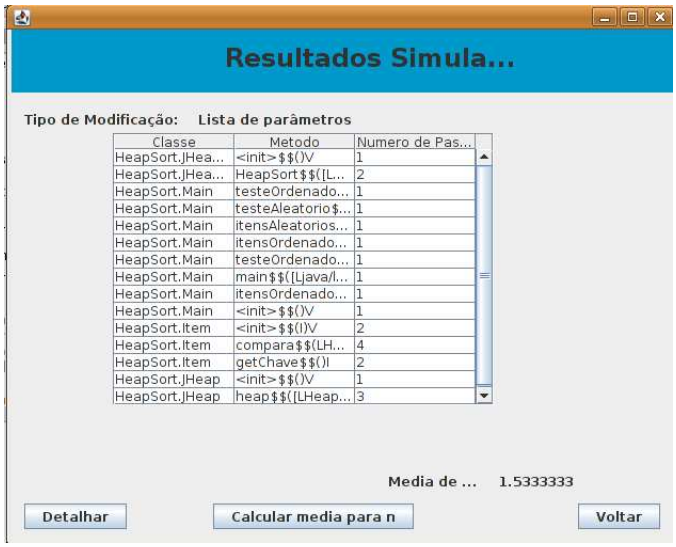

Figura 16: Interface de usuário *Exibir Resultado da Consulta*

- *Classes de negócio:* contém as classes necessárias para coleta de dados para simulação e para a realização dos caculos necessários.
- *Classes de Persistência:* contém as classes responsáveis pelo armazenamento e recuperação de dados.

Na Tabela 21 podemos ver os objetivos de todas as classes do sistema.

4.3.1 Implementação

O *Simulador* foi desenvolvido em linguagem Java, utilizando para tal a ferramenta *IDE NetBeans* Versão 6.2. Possui um total de 17 classes perfazendo um total de 1300 linhas de código.

O funcionamento da ferramenta baseia-se na análise do código das classes de um sistema. Para garantir a portabilidade da ferramenta em relação às diferentes versões de Java, essa análise é realizada diretamente no *bytecode* das classes, ao invés de analisar o código fonte diretamente. Dessa maneira contou-se com o auxilio da biblioteca BCEL *(Byte Code Engineering Library)* , que fornece recursos para análise de arquivos bytecode de Java.

O ponto central da ferramenta é a modelagem da conexão entre as classes do sistema, de maneira a ser feita de acordo com o tipo de ligação entre elas. Isso porque, dependendo do tipo de modificação a ser simulada, é necessário interligar as classes de maneira diferente. Desse modo, na maioria dos casos, o sistema sendo analisado pode ser interpretado como um Grafo Direcionado, no qual:

- um nó representa uma classe do sistema,
- uma aresta de A para B representa uma conexão na qual A é a classe usuária de B.

4.3.2 Algoritmo para Simulação de Modificação em Métodos

A estrutura usada para a coleta dos dados necessários no Software para a simulação de modificações em métodos, foi um *HashMap.* com o auxilio de alguns dados encapsulados por

Nome da Classe	Objetivo
InterfaceMapMetodos	Encapsular as Interfaces implementadas por uma classe e seus respectivos métodos
jfrmDetalhesSimulacaoMetodos	Classe de apresentação de detalhes da simulação de um método
jfrmEscolheArquivo	Classe de apresentação que permite a seleção de arquivos para simulação
jfrmEscolherAlteracao	Classe de apresentação para escolha do tipo de modificação a ser simulada
jfrmModificacoesMetodos	Classe de apresentação para especificação de alteração em métodos
jfrmResultadoMetodos	Classe de apresentação para exibição da simulação em métodos
jfrmAbrirArquivo	Classe de apresentação que permite a seleção de um arquivo para consulta de dados
jfrmCalculon	Classe de apresentação para calculo de média de passos para um valor diferente de n
jfrmColetaAutomatica	Classe de apresentação para coleta automática de dados
jfrmMostrarHist	Tela de apresentação para exibição de dados de simulação já realizada
jfrmTelaInicial	Tela de apresentação de interface principal da ferramenta
LerHistorico	Classe responsável por recuperar dados salvos de simulações anteriores
ModeloHistorico	Classe que contém os dados da simulação a serem armazenados ou recuperados
ReferenciaAMetodos	Classe de negócio responsável pela coleta das referências de métodos
ReferenciasEClasses	Classe de encapsulamento responsável pelo armazenamento do nome do método e a classe a qual pertence
SalvarHistorico	Classe responsável por salvar dados de uma simulação
SimulacaoMetodos	Classe de negócio responsável pela execução da simulação em métodos

Tabela 21: Objetivos das classes do *Simulador*

outras classes do *Simulador*. Tal simulação possui dois critérios de modificação, para os quais foram implementados algoritmos distintos de simulação.

Nos *pseudocodigos* a seguir, consideraremos:

1. M = método que será renomeado.
2. classesVisitadas = conjunto que armazena o nome das classes que usam M.
3. metodosVisitados = conjunto que armazena os métodos e sua respectiva classe que utilizam M.
4. passos = número de passos para renomear M.

A seguir são descritos os tipos de moficações possíveis, quando se trata de métodos, e seus respectivos algoritmos para simulação:

- **Renomear Método.** Para simular a alteração do nome de um método, não é necessário que os outros métodos que o chamem também sofram alteração, logo, precisamos apenas identificar as classes nas quais exista uma ocorrência do método em questão, sendo a quantidade dessas classes, o número de passos para realizar a modificação.

```
1  passos = contaPassos(M);
2
3  int contaPassos(metodo M)
4
5  inicio
6    int passos = 1;
7    inserir a classe CM do metodo M em classesVisitadas
8
9    para cada classe C que usa M faca
10         passos++;
11         se C nao esta em classesVisitadas entao
12                 inserir C em classesVisitadas
13         fim se
14
15         para cada metodo X em C que usa M faca
16                 inserir X em metodosVisitados com sua respectiva classe C
17         fim para
18   fim para
19   retorne passos;
20 fim
```

- **Lista de parâmetros.** Para simular a alteração da lista de parâmetros de um método, consideramos que será inserido um novo parâmetro. Se um método *M2* chamar o método sendo alterado *M1*, é possível que *M2* também tenha sua lista de parâmetros alterada. Isso ocorre porque existem duas possibilidades para que *M2* forneça o parâmetro necessário para *M1*, sendo elas:

  1. o próprio *M2* produzirá o parâmetro para *M1*
  2. *M2* necessitará receber o parâmetro como um dado para, então, passar para *M1*

Como não é possível definir automaticamente quando *M2* será capaz de produzir os dados para *M1*, para fins de simulação de propagação desse tipo de modificação, consideramos o pior caso, ou seja, o Caso 2. Dessa maneira, a propagação desse tipo de modificação seria contada até que se chegasse a um método que não é chamado por nenhum outro no programa, sendo a pior hipótese, o método *main*. Para amenizar essa premissa pessimista, consideramos como ponto de parada da simulação, o método que estiver definido em uma interface implementada pela sua classe, sendo que essa interface não poderá sofrer alterações.

Sendo assim, qualquer método *M* que chamar o método *M1* e não estiver nas interfaces de parada, também sofrerá os impactos da inclusão de um novo parâmetro, tornando o algoritmo recursivo.

```
1  passos = contaPassos(M);
2
3  int contaPassos(metodo M)
4
5  inicio
6    passos = 0;
7
8    se M pertence a alguma interface de parada entao
9          retorne passos;
10   fim se
11
12   para cada classe C que usa M faca
13
14         se C nao e a propria classe de M entao
15
16           se C ainda nao foi visitada entao
17                 passos++;
18                 inserir C em classesVisitadas
19       fim se
20
21       para cada metodo X em C que usa M faca
22
23                 se X nao estiver nas interfaces de parada faca
24                   se X ainda nao foi visitado faca
25                         inserir X em metodosVisitados
26                         passos += contaPassos(X);
27                   fim se
28             fim se
29           fim para
30
31           senao
32                 para cada matodo X em C que usa M faca
33
34                   se X n o estiver nas interfaces de parada entao
35                         se X ainda nao foi visitado entao
36                                 inserir X em metodosVisitados
37                                 passos += contaPassos(X)
```

```
38                              fim se
39                          fim se
40                      fim para
41              fim senao
42      fim para
43      retorne passos;
44  fim
```

4.4 Conclusão

Nessa seção descrevemos a ferramenta *Simulador*, que realiza a simulação de alterações em *software* a nível de código fonte, sendo implementada na linguagem Java. A ferramenta apresenta relatórios dos resultados obtidos pela simulação, assim como promove o armazenamento e a consulta dos dados gerados.

5 Conclusão

O presente relatório mostrou o trabalho desenvolvido durante o período de Iniciação Científica. As principais contribuições desse trabalho, além da oportunidade que tive de fixar meus conhecimentos na rea de Engenharia de Software, foram:

1. Implementação do módulo na ferramenta CONNECTA que permite a coleta automática de dados.

2. Implementação do módulo na ferramenta CONNECTA que permite o recálculo da métrica *KB3* via a inserção de novos valores paras as contantes de acoplamento utilizadas no caculo da mesma.

3. Implementação da ferramenta de simulação de modifição em código fonte de *softwares* feitos utilizando a linguagem Java.

6 Referências Bibliográficas

1. FERREIRA, K. A. M. Avaliação de Conectividade em Sistemas Orientados por Objetos. Dissertação (Mestrado) Departamento de Ciência da Computação / Universidade Federal de Minas Gerais, Belo Horizonte, 2006.

2. FERREIRA, K. A. M. Um Modelo de Prediçõ de Amplitude da Propagao de Modificações Contratuais em Software Orientado por Objetos. Dissertaçõ (Doutorado) Departamento de Ciência da Computaçõ / Universidade Federal de Minas Gerais, Belo Horizonte, 2011.