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

Heitor Corrêa de Almeida
Orientadora: Profa. Dra. Mariza Andrade da Silva Bigonha

Projeto Fapemig - 2007-2009
Connecta: Conectividade em Módulos
Reestruturação de Connecta
e
Implementação, Experimentos e Análise da Métrica Coesão de Interesses

Belo Horizonte – MG
2009 / 1º semestre

Sumário

Lista de Figuras ..... **iv**

Lista de Tabelas ..... **v**

Lista de Siglas ..... **vi**

1 Introdução **7**

2 Métricas de Coesão **9**

2.1 Ausência de Coesão em Métodos 10

2.2 Contrato e Coesão de Interesse 11

2.3 Implementação da Métrica de Coesão de Interesse 11

2.3.1 Algoritmo 12

2.3.2 Correções 13

2.3.3 Testes 14

2.4 Conclusão 15

3 Análise Comparativa entre as Métricas de Coesão de Interesse e LCOM **16**

3.1 Metodologia 16

3.2 Testes e Análise dos Resultados 17

3.3 Formato dos Experimentos para a Avaliação Comparativa entre as Métricas Coesão de Interesse e LCOM 17

3.4 Classes de CONNECTA 18

3.4.1 RealizaConsulta.java 18

3.4.2 ClassModule.java 19

3.4.3 ConnectionPath.java 20
3.4.4 frmClassDetail.java 22
3.4.5 frmSelectFiles.java 23
3.5 Classes Desenvolvidas por Alunos Iniciantes em OO 25
3.5.1 Professores.java 25
3.5.2 Main.java 26
3.6 Resultados 30
3.7 Avaliação Crítica 30
3.7.1 Falhas da Métrica LCOM 31
3.7.2 Falhas da Métrica Coesão de Interesse 32
3.8 Aperfeiçoamento da Métrica de Coesão de Interesse 32
3.8.1 Motivação 32
3.8.2 Ajuste 33
3.9 Conclusões 34

4 Reestruturação de CONNECTA 35

4.1 Implementação de Uma Nova Forma de Execução para CONNECTA 35
4.1.1 Solução Adotada 36
4.1.2 Implementação 36
4.2 Implementação de Suporte a Banco de Dados 37
4.2.1 Gerenciador de Banco de Dados MySQL 37
4.2.2 Implementação 37
4.2.3 Desenvolvimento dos Scripts 38
4.2.4 Desenvolvimento do Código 39
4.3 Conclusão 41

5 CONCLUSÃO 42

Referências Bibliográficas **43**

Lista de Figuras

Figura 4.1 Esquema Relacional para o Banco de Dados de CONNECTA 38

Lista de Tabelas

Tabela 2.1 Software para Testes 14

Tabela 3.1 Resumo das Análises 30

Lista de Siglas

OO	*Orientação por Objetos, Orientado por Objetos*
LCOM	*Lack of Cohesion in Methods, Ausência de Coesão em Métodos*
IDE	*Integrated Development Environment, Ambiente Integrado para Desenvolvimento de Software*
CoIn	*Coesão de Interesse*
TAD	*Tipo Abstrato de Dados*
SGBD	*Sistema de Gerenciamento de Bancos de Dados*
API	*Application Programming Interface*

1 Introdução

Este documento descreve as atividades realizadas pelo autor, aluno de graduação do curso de Ciência da Computação da Universidade Federal de Minas Gerais, em seus trabalhos de iniciação científica no projeto financiado pela FAPEMIG intitulado Connecta: Conectividade em Módulos, durante o período de agosto de 2008 a dezembro de 2009, sob a orientação da professora Mariza Bigonha e da aluna de doutorado Kecia Aline Marques Ferreira.

O principal foco do trabalho é um software denominado CONNECTA, desenvolvido por Ferreira (1) em sua dissertação de mestrado. CONNECTA obtém os valores de diversas métricas relacionadas à avaliação de conectividade e manutenibilidade a partir do *bytecode* de programas escritos na linguagem Java. As atividades aqui descritas consistem, essencialmente, de aprimoramentos realizados na ferramenta, com o objetivo de torná-la mais robusta, completa, facilitar sua execução e atestar sua confiabilidade. As atividades realizadas são agrupadas de acordo com as seguintes grandes metas: implementação da métrica de coesão de interesse, análise comparativa entre métricas de coesão, aperfeiçoamento da nova métrica, implementação de uma nova forma de execução do CONNECTA e implementação e incorporação de um banco de dados nessa ferramenta.

Cada uma das seções seguintes deste relatório dedica-se à explanação dos contextos, objetivos, procedimentos e resultados da respectiva grande meta.

Na Seção 2 é descrita a implementação de uma nova métrica para avaliar coesão em software orientado por objetos, denominada Coesão de Interesses(CoIn).

A Seção 3 faz uma análise comparativa entre a métrica de interesse e a métrica LCOM, já disponível em CONNECTA, ambas com o mesmo fim. Essas métricas avaliam quão relacionados estão os elementos internos dos módulos que compõem um programa. Quanto maior a coesão interna dos módulos, menor o acoplamento e a complexidade do sistema. Mensurar a coesão é, portanto, tarefa de extrema relevância na confecção de programas que venham a demandar custos mínimos com manutenção. Boa manutenibilidade é desejável uma vez que a manutenção corresponde à maior fatia do custo total de um sistema. A Seção 3.8 mostra o

aperfeiçoamento feito na métrica de coesão de interesse. Os testes realizados durante a análise comparativa evidenciaram uma falha da métrica recém-implementada e indicaram o caminho para sua correção.

A Seção 4 mostra a reestruturação de CONNECTA. Especificamente, a Seção 4.1 mostra a implementação de uma nova forma de execução. A maneira de ativar o programa, até então utilizada, não era suficientemente prática, além de exigir conhecimentos que provavelmente escapariam ao usuário comum. Decidiu-se, por esse motivo, estabelecer uma nova forma de executá-lo. Foi desenvolvido um outro sistema, de mais alto nível e caracterizado por uma alta usabilidade, cuja função é encapsular ao máximo os detalhes complexos dessa execução e realizá-la. Destaca-se também nessa seção a implementação do suporte a banco de dados. Os valores para as métricas, obtidos utilizando-se o CONNECTA, podiam ser armazenados apenas em arquivos, caso o usuário quisesse manter os registros. O armazenamento de grandes quantidades de informação, porém, é mais eficiente, organizado e confiável quando apoiado por um sistema de banco de dados.

A Seção 5 conclui este relatório apresentando as contribuições do nosso trabalho e possíveis trabalhos futuros.

2 Métricas de Coesão

Métricas (1), no contexto da Ciência da Computação, são padrões de medição para avaliar aspectos que de alguma forma caracterizam um programa, suas especificações ou seus processos de produção e que podem ser mensurados quantitativamente, com a atribuição de valores que possibilitam estabelecer comparações. A avaliação de métricas de software torna-se essencial no contexto da engenharia de software, universo altamente competitivo e capitalista, na medida em que possibilita, dentre outras vantagens, a redução de custos. De posse de instrumentos de medida, pode-se avaliar, controlar, corrigir e aprimorar propriedades específicas de um programa.

Coesão, como métrica de qualidade de software, foi definida inicialmente por Myers et al. em (2) baseada em boas práticas de programação que reduziam custos de modificação e, consequentemente, custos de manutenção. Contudo, essa medida foi estabelecida de forma ordinal, o que significa que classifica em uma escala de níveis e não em medidas quantitativamente precisas. Isso abriu caminho para que muitas outras métricas de medição de coesão fossem adotadas((3), (4), (5), (6)).

Coesão, segundo Myers (2), é o grau de relacionamento entre os elementos internos de um módulo. Também definida como a medida de quão fortemente relacionadas e concentradas estão suas várias responsabilidades, está intimamente relacionada à qualidade e aos custos do software, mais especificamente ao custo de manutenção. A manutenibilidade depende de um bom grau de coesão. Um alto grau de coesão implica em um baixo grau de acoplamento entre módulos, o que torna mais ágil a realização de quaisquer mudanças no código. Em software orientado por objetos(OO), definimos módulo como uma classe, composta por atributos e métodos. Por conseguinte, a coesão interna de uma classe é definida como o grau de interrelacionamento entre seus atributos e métodos. Módulos com uma classificação de coesão alta costumam ser preferíveis pois alta coesão está associada a diversas outras características desejáveis de software, como robustez, confiabilidade, reusabilidade e compreensibilidade, enquanto baixa coesão se associa a características indesejáveis como, por exemplo, dificuldade de realizar testes, baixa reusabilidade e dificuldade de compreensão.

Em vista da relevância da coesão na determinação dos custos de manutenção que, segundo Meyer (7), ultrapassam 70% do custo total de um sistema, torna-se interessante estabelecer alguma forma de avaliá-la. Para tanto, é preciso definir métricas adequadas. Dentre as 20 diferentes métricas propostas na literatura para avaliação de coesão interna de módulos (3), (4), (5), (6), a mais popular delas em relação à OO é a *Lack of Cohesion in Methods*(LCOM) (3), muito embora críticas plausíveis possam ser feitas à sua avaliação. Na Seção 3 mostramos, por meio de testes realizados, que essa métrica segue um método de análise bastante questionável, sujeito a determinadas falhas que, em alguns casos, comprometem de forma crítica a avaliação.

O questionamento da validade da métrica LCOM como avaliadora de coesão interna em um módulo de software OO, sendo ela a métrica mais popularmente aceita com esse objetivo, pode ser tomado como estímulo para o desenvolvimento de uma métrica diferente para avaliação do mesmo aspecto.

Ferreira et al. (8) propõem uma métrica para avaliar coesão baseada em contratos da classe, intitulada Coesão de Interesse(CoIn). Essencialmente, essa métrica realiza a avaliação de um software com base no número de contratos estabelecidos por uma classe. CoIn foi recentemente implementada e incorporada à ferramenta CONNECTA (1), pelo autor deste documento, e será descrita na Seção 2.2.

2.1 Ausência de Coesão em Métodos

LCOM(ausência de coesão em métodos) é uma das métricas do conjunto proposto por Chidamber e Kemerer (3) para avaliação de software orientado por objetos, um dos mais referenciados na literatura. Os autores interpretaram a coesão interna de um módulo como a coesão entre os métodos de uma classe e sua observação é realizada com base na similaridade entre eles. Métodos similares são métodos que usam variáveis de instância de classe em comum.

Definição de LCOM conforme (1): seja P o conjunto formado pelos pares de métodos que não possuem variáveis de instância em comum e Q o conjunto formado pelos pares de métodos que possuem variáveis de instância em comum. Se nenhum método da classe utiliza variáveis de instância da classe, P é vazio . O cálculo de LCOM é dado por:

$LCOM = |P| - |Q|$, se $|P| > |Q|$
$LCOM = 0$, caso contrário

2.2 Contrato e Coesão de Interesse

Para estabelecer a definição de Coesão de Interesse (8), a métrica cuja implementação e avaliação constitui parte deste trabalho, é preciso primeiro definir o conceito de interesse. Um interesse é um conjunto de operações relacionadas. Para benefício da modularidade, é importante a criação de classes que implementam um único interesse. Um exemplo de composição de interesses é a implementação, em uma única classe, dos tipos abstratos de dados fila e pilha. Essa classe implementa dois interesses. Seu grau de coesão é inferior ao de outras duas classes que implementam, cada uma, apenas o tipo fila ou o tipo pilha, classes de um único interesse.

A métrica Coesão de Interesse baseia-se no número de interesses que uma classe implementa. Seu valor é então dado por $1/$*total de interesses da classe* . Se uma classe implementa dez interesses, por exemplo, o valor assumido pela métrica é 0.1. Da mesma forma, em uma classe que implementa apenas um interesse a métrica assume o valor 1. Ou seja, quanto mais próximo de 1 é o valor da métrica, melhor a sua coesão de interesse. Quanto mais próximo de 0, pior. Para definir a métrica com mais precisão, os seguintes conceitos se fazem necessários (8):

Relacionamento: dois métodos de uma classe C estão relacionados se utilizam pelo menos um atributo da classe C em comum ou pelo menos um método da classe C em comum.

Propriedade Transitiva: se um método **a** está relacionado a um método **b** pela definição de relacionamento anterior, e **b** está relacionado a um método **c**, então o método **a** está também relacionado a **c**.

Interesse: os conjuntos disjuntos formados por todos os métodos relacionados entre si de uma classe compõem os interesses dessa classe. Ou seja, um interesse é um conjunto de métodos relacionados.

Definição de CoIn: seja C o conjunto de conjuntos disjuntos formados por métodos com relacionamento entre si. Seja o número de interesses coesos $N = |C|$.

$Coesao = 1/N$, se $N > 0$
$Coesao = 0$, caso contrário

2.3 Implementação da Métrica de Coesão de Interesse

A métrica CoIn foi implementada em CONNECTA pelo autor, com base em sua definição, dada na Seção 2.2. Para implementá-la, em primeiro lugar, projetamos um algoritmo

descrito em alto nível na próxima seção e realizamos algumas correções no sistema.

Nesse algoritmo é utilizada uma função hash. Isso significa que realizamos um mapeamento de valores de entrada da função para valores de saída da função. Isso se dá na medida em que mapeamos, para determinado valor chave, um determinado conjunto de valores relacionados. A chave, no caso, é o número que identifica um método, enquanto os valores relacionados são as referências encontradas no mesmo método.

2.3.1 Algoritmo

```
1. Obtém um mapeamento de métodos para suas referências a campos e
   métodos da classe.

2. Inaugura-se um primeiro interesse com as referências do primeiro
   método.

3. Para cada chave desse mapeamento hash (para cada método), a partir
   da segunda (chave A):
    3.1. Para cada chave anterior (chave B):
        3.1.1. Obtém interseção entre os conjuntos de referências
               mapeadas para as duas chaves.
        3.1.2. Se a interseção não foi vazia:
            3.1.2.1. Se é a primeira interseção não-vazia encontrada,
                     então chave A entra no interesse de B.
            3.1.2.2. Senão:
                3.1.2.2.1. Número de interesse de B é sobrescrito por
                           número de interesse de A.
                3.1.2.2.2. Para toda chave (chave C):
                    3.1.2.2.2.1. Se número de interesse de C é igual ao
                                 número recém-sobrescrito de interesse
                                 de B, então número de interesse de C é
                                 sobrescrito por número de interesse de A.
        3.1.3 Se interseção foi vazia para todas as chaves anteriores,
              inaugura novo interesse.
```

2.3.2 Correções

Antes mesmo da implementação da métrica CoIn, algumas pequenas incoerências já podiam ser ocasionalmente observadas nos relatórios fornecidos pelo CONNECTA. Após a inserção da métrica no sistema, entretanto, essas incoerências tornaram-se mais evidentes e receberam a devida atenção. Esta seção dedica-se à explanação dos problemas encontrados e soluções aplicadas a eles.

access$

Ao relatar os valores de LCOM e de CoIn, para algumas classes, o programa apresentava valores inconsistentes e, portanto, incorretos, como número de interesses maior até que o número total de métodos.

Revisando-se o código, nenhum problema foi encontrado. Então, usou-se a estratégia de se imprimir os nomes de cada método à medida em que eles eram inspecionados pelos métodos analisadores de *bytecode* de CONNECTA. Após vários experimentos, constatou-se que no *bytecode* de classes que continham funções de interface gráfica haviam mais métodos do que no código original da classe, em Java. Provavelmente eram métodos adicionados na conversão para *bytecode*. Os nomes dos métodos adicionais encontrados, felizmente, seguiam um padrão: a string "access$" seguida de alguma centena (000, 100, 200, 300 etc).

O padrão seguido pelo nome dos métodos extra foi utilizado na resolução do problema. Assim, logo na primeira análise dos *bytecodes* de métodos, o que constitui, em CONNECTA, um passo tanto da coleta da métrica LCOM quanto da coleta da métrica Coesão de Interesse, métodos possuindo um nome iniciado por "access$" são sumariamente ignorados. Ignoramos assim suas referências a campos e métodos da classe.

Após a aplicação dessa abordagem o problema foi sanado, mas a um custo: todo método que tenha seu nome iniciado por "access$" será ignorado na análise, independentemente de ter sido adicionado na conversão para bytecode ou na confecção do código original em Java, o que significa que a avaliação de Connecta será inválida caso o usuário teste classes que contenham algum método cujo nome é iniciado por essa string.

Adição à Interface Gráfica

Após a implementação da métrica CoIn, era necessário adicioná-la à interface gráfica. Isso foi feito adicionando-se variáveis à classe *frmClassDetail.java*, por meio de recursos fa-

cilitadores oferecidos pela IDE NetBeans e que permitem alterar a aparência da interface com cliques do mouse enquanto o código é gerado automaticamente. Além disso, foi preciso adicionar um novo campo e novos métodos à classe *ModelClassHistory.java* e novas chamadas de métodos na classe *frmResultSystem*.

2.3.3 Testes

O objetivo de se realizar os testes é obter valores referenciais para as métricas. No caso da recém criada métrica de Coesão de Interesse, são desconhecidos valores médios que poderiam ser tomados como satisfatórios. Assim, neste sentido, testes foram realizados com um amplo conjunto de produtos de software livre pois, além de possibilitarem acesso ao *byte-code* de suas classes, o que é um pré-requisito para a submissão ao Connecta, programas dessa categoria comumente possuem maior qualidade de código, o que implica na obtenção de valores almejáveis para as métricas.

Programas para diversos fins foram analisados. A seguir, a Tabela 2.1 mostra as áreas de aplicação e os sistemas avaliados.

Área	Software
Clustering	Essence
	Gridsim
	JavaGroups
	Prevayler
	Super
DataBase	DBUnit
	Exist
	Hibernate
	Squire
Desktop	Facilitator
	JavaGuiBuilder
	JavaX11Library
	Jpilot
	lobo
	Scope
Development	CodeGenerationLibrary
	DrJava
	FindBugs
	JasperReports
	Junit
	lobo
	SpringFramework

Tabela 2.1: Software para Testes

Os resultados obtidos com as análises dos programas podem ser encontrados no artigo publicado (9).

2.4 Conclusão

Esta seção tratou da realização, com sucesso, da implementação de uma nova métrica avaliadora de coesão, Coesão de Interesse, proposta por Ferreira et al. (8), com o objetivo de expandir o ferramental disponível para avaliação de coesão em software. Muitas métricas já foram propostas para tal fim, mas suas avaliações são questionáveis e, em muitos casos, inapropriadas, o que basta para o surgimento da necessidade de criação de novas métricas substitutas ou, pelo menos, complementares. Além disso, relatou a realização de um conjunto de testes para obtenção de valores referenciais para as métricas coletadas por CONNECTA.

3 Análise Comparativa entre as Métricas de Coesão de Interesse e LCOM

O trabalho apresentado nesta seção baseou-se em estudos teóricos e experimentais, envolvendo o planejamento, execução e análise de experimentos, que demandaram obtenção e aplicação de conhecimentos por parte do autor a respeito do que caracteriza uma classe com boa coesão.

3.1 Metodologia

Os valores para as métricas, cuja observação nos permite compará-las, foram obtidos por meio da aplicação da ferramenta CONNECTA.

Um problema recorrente na área de medição de software é a obtenção de dados reais e industriais para a realização dos experimentos, pois a coleta de métricas envolve, muitas vezes, a análise de código fonte ou compilado dos programas. Uma alternativa para isso é analisar dados de software livre. Sourceforge (10), por exemplo, é um *website* que reúne milhares deles. Em estudos anteriores realizados com a ferramenta CONNECTA, foram obtidos e utilizados com sucesso vários programas de código aberto disponíveis nesse endereço. Pretendia-se, neste trabalho, novamente utilizá-lo como fonte de *bytecodes*. Porém, um empecilho surgiu à utilização desses programas: uma avaliação não automatizada e consistente de coesão poderia requerer do avaliador o conhecimento da forma como cada classe avaliada se integra a todo o sistema. O avaliador deveria conhecer, de forma relativamente profunda, a estrutura de todo o código do software. Assim, para que esse pré-requisito fosse atendido, escolheu-se utilizar, para a bateria inicial de testes, o código fonte do próprio CONNECTA, já que o avaliador o teria conhecido a fundo durante a implementação da métrica CoIn.

3.2 Testes e Análise dos Resultados

O texto a seguir descreve um conjunto de testes que visam comparar as avaliações de coesão realizadas segundo as métricas LCOM e Coesão de Interesse. As métricas foram implementadas no sistema CONNECTA e as classes utilizadas na Seção 3.4 pertencem ao próprio programa. Essas classes foram selecionadas por constituírem exemplos de módulos não muito longos, adequados para uma avaliação humana, não automatizada, da coesão. O objetivo de tal avaliação é construir uma base para a análise crítica dos resultados das duas métricas. As duas últimas classes avaliadas, apresentadas na Seção 3.5, pertencem a um sistema para administração simplificada de uma universidade, e foram desenvolvidas por alunos iniciantes na programação OO.

3.3 Formato dos Experimentos para a Avaliação Comparativa entre as Métricas Coesão de Interesse e LCOM

Para cada uma das classes utilizadas, o seguinte formato padrão é aplicado na realização e descrição dos experimentos:

Nome da Classe: o nome do arquivo *bytecode* Java, gerado a partir do código fonte.

Listagem de Métodos: descrição da funcionalidade de cada método pertencente à classe.

Avaliação Geral de Coesão: avaliação que leva em conta o conceito original de coesão, sem se ater a uma métrica específica, visando obter um possível ponto de referência para contrastar os resultados fornecidos pelas métricas.

Avaliação segundo LCOM: valor de coesão obtido de acordo com a métrica LCOM.

Avaliação segundo a Coesão de Interesse: valor de coesão obtido de acordo com a métrica Coesão de Interesse.

Análise Comparativa: comparação dos valores obtidos à avaliação geral. Destaque de motivos aos quais atribuir o sucesso ou insucesso das avaliações específicas.

3.4 Classes de CONNECTA

3.4.1 RealizaConsulta.java

- Construtora `RealizaConsulta`: utiliza todos os atributos da classe, para tornar a instância pronta para acessar o banco de dados. Realiza uma sequência de passos para tornar um objeto, atributo da classe, pronto e disponível para utilização no acesso ao banco de dados.

- Método `Consulta`: utiliza todo o serviço do método construtor por fazer uso do objeto, atributo da classe, por ele inicializado. O objetivo desse método é realizar uma consulta ao banco de dados retornando um conjunto de resultados.

- Método `Atualiza`: utiliza todo o serviço do método construtor por utilizar o objeto, atributo da classe, por ele inicializado. O objetivo desse método é realizar uma alteração na coleção de dados do banco retornando o número de linhas alteradas.

- Método `Fecha`: caso a instância ainda não tenha sido fechada, fecha conexão com o banco de dados, encerrando objetos iniciados na construtora. Isso se faz necessário para o estabelecimento de novas conexões em novas instâncias.

- Método `Finalize`: se assegura do correto encerramento da instância de RealizaConsulta chamando o método Fecha. É chamado pelo coletor de lixo na finalização que este realiza automaticamente.

Avaliação Geral de Coesão

O método construtor está fortemente relacionado a todos os outros métodos, por inicializar tudo o que os outros utilizam. Consulta, Atualiza e Fecha implementam diferentes operações, mas utilizam uma base comum. Considerando-se o fato de se apoiarem sobre o mesmo conjunto de dados comuns, não há falha alguma de coesão. Por esse motivo, a classe possui alto grau de coesão interna. Segundo a classificação de Myers, para módulos, a coesão identificada seria a coesão informacional, segundo maior nível de coesão. De acordo com a classificação de coesão para classes, a coesão identificada seria a coesão contratual, o maior nível de coesão. O método Finalize apenas evoca o método Fecha e, por ser nada mais que uma função "destrutora", que libera os recursos alocados pela instância, em nada deveria contribuir para a redução ou aumento da coesão.

Avaliação segundo LCOM

Valor 0. Melhor valor de coesão que pode ser obtido com a métrica. Ocorre porque o

número de pares de métodos que utilizam campos da classe em comum é maior que o número de pares de métodos que não utilizam campos da classe em comum.

Avaliação segundo a Coesão de Interesse

Valor 0,5. Segundo melhor valor de coesão que pode ser obtido com a métrica. Ocorre porque o método Finalize foi considerado como um interesse à parte da classe, por não utilizar campos ou métodos da classe utilizados por outros métodos da classe.

Análise Comparativa

Ambas as métricas ofereceram valores coerentes com a Avaliação Geral. Porém LCOM obteve um valor mais próximo do que se poderia esperar. Tendo detectado dois interesses, a métrica Coesão de Interesse exibiu uma certa imprecisão em sua avaliação.

3.4.2 ClassModule.java

- Construtora `ClassModule`: inicializa todos os campos da classe, inclusive seu objeto do tipo ClassCollector. Todos os campos contêm, essencialmente, informações de interesse sobre um módulo.
- Método `getCoesaoContratual`: apenas retorna o valor da métrica Coesão Contratual da classe representada por esse módulo. Valor devidamente obtido durante a sequência construtora.
- Método `getClassPath`: apenas retorna a string devidamente inicializada na construtora.
- Método `getName`: apenas retorna nome da classe representada por esse módulo.
- Método `getIndex`: apenas retorna o índice da classe representada por esse módulo.
- Método `toString`: retorna uma string conjunta com índice, nome e caminho da classe representada.
- Método `getExpectedChange`: apenas retorna o valor do campo expectedChange do módulo.
- Método `getZeroChange`: apenas retorna o valor do campo zeroChange do módulo.
- Método `setExpectedChange`: atribui ao campo expectedChange o valor recebido como parâmetro.
- Método `setZeroChange`: atribui ao campo zeroChange o valor recebido como parâmetro.

Avaliação Geral de Coesão

Pode-se enxergar um forte relacionamento entre todos os métodos por seguirem a premissa básica de oferecer ou determinar informações sobre o módulo, ou melhor, sobre a classe representada pelo módulo. Cada método realiza uma função diferente e independente, mas o objetivo de todos é basicamente o mesmo. As responsabilidades estão relacionadas e concentradas. Tais observações sugerem um alto grau de coesão. A classificação segundo a coesão interna de classes seria a coesão identificada como coesão comunicacional, segundo maior grau de coesão. O único impedimento para um nível máximo de coesão é a existência de um campo público na classe.

Avaliação segundo LCOM

Valor 17. Considerando que temos 10 métodos na classe, o valor máximo que LCOM poderia obter, indicando o pior nível possível de coesão, é a combinação de dez, dois a dois, que é dada por 10! / (2! * (10-2)!) que é igual a 45. Ou seja, LCOM forneceu 17 em 45, o que equivale a 38% de "não-coesão".

Avaliação segundo a Coesão de Interesse

Valor 1. Melhor valor possível de coesão utilizando-se essa métrica.

Análise Comparativa

A avaliação LCOM forneceu um valor incoerente com a Avaliação Geral, indicando uma grande falta de coesão, enquanto Coesão de Interesse demonstrou uma avaliação muito mais próxima do esperado, quase perfeitamente coerente com o que se pôde observar na avaliação geral. Ressalta-se dessa forma uma falha da métrica LCOM.

3.4.3 ConnectionPath.java

- Construtora `ConnectionPath`: apenas inicializa uma pilha de conexões para outros módulos, objeto atributo da classe.
- Método `isEmpty`: informa se a pilha de conexões está vazia ou não.
- Método `insertElement`: empilha uma conexão passada como parâmetro.
- Método `removeElement`: desempilha uma conexão, caso exista alguma a ser desempilhada, e a retorna.
- Método `getFirstElement`: retorna a conexão do topo da pilha, sem desempilhá-la.
- Método `printPath`: possui apenas uma impressão no terminal. Porém, como se encontra comentada, temos um método vazio.

- Método `clear`: limpa a pilha, removendo todos os seus elementos.
- Método `copy`: cria uma nova instância da classe, cuja pilha de conexões é igual à da atual instância, e a retorna.
- Método `getProbability`: calcula a probabilidade, atributo da classe, que é o produto dos pesos das conexões da pilha de conexões.
- Método `isInPath`: verifica se determinada instância de ClassModule, passada como parâmetro, está em alguma das conexões empilhadas.
- Método `clearSubPath`: apenas desempilha uma conexão.

Avaliação Geral de Coesão

Há um forte relacionamento entre praticamente todos os métodos por estabelecerem formas de manipulação e/ou fornecerem informações a respeito da mesma pilha de conexões. Os métodos insertElement, removeElement, clear, copy e clearSubPath são todos de manipulação da pilha, enquanto isEmpty, getFirstElement, copy, getProbability e isInPath são métodos para obtenção de informações a respeito dela. Há apenas uma ressalva a respeito do método printPath. É um método que se encontra vazio. Não tem sua função relacionada às funções dos outros métodos e por isso prejudica a coesão. Ainda sim, por estarem todos os outros focados para um mesmo fim ou, no máximo, dois fins semelhantes, temos um alto grau de coesão na classe.

Avaliação segundo LCOM

Valor 0. Máximo valor de coesão. Significa que existem, na classe avaliada, mais pares de métodos que se relacionam do que pares de métodos que não se relacionam por meio do uso em comum de atributos da classe.

Avaliação segundo a Coesão de Interesse

Valor 0,5. Segundo maior valor de coesão. Significa que, de acordo com a definição de interesse estabelecida pela métrica, foram encontrados dois interesses na classe avaliada.

Análise Comparativa

Ambos os valores fornecidos pelas avaliações anteriores das métricas são coerentes com a observação geral da existência de um alto grau de coesão. Porém, a métrica Coesão de Interesse foi capaz de perceber e avaliar essa coesão com uma precisão superior à métrica LCOM. O método vazio não tem relacionamento algum com os outros e, portanto, deveria prejudicar a coesão da classe. Como LCOM indicou a falta de coesão mínima, ou seja, a coesão máxima, sua avaliação desconsiderou o fato.

3.4.4 frmClassDetail.java

- Construtora `frmClassDetail`: inicializa quase todos os campos da classe e os elementos da interface. Como esta classe fornece detalhes a respeito de uma determinada classe, evoca uma série de métodos posteriormente declarados e definidos para permitir o acesso a essas informações. Além disso, determina o texto de dois rótulos da interface.

- Método `setClassModule`: chamado pela construtora, determina como a classe representada pela instância desta classe, aquela cujo nome é passado como parâmetro.

- Método `showClassMetrics`: chamado pela construtora, determina o texto de diversos rótulos da interface com informações da classe representada pela instância.

- Método `showEmptyData`: chamado pela construtora caso a classe que deveria ser representada pela instância não tenha sido encontrada e, consequentemente, suas informações não tenham sido obtidas. Determina textos vazios para os rótulos da interface.

- Método `listConnections`: chamado pela construtora. Determina as linhas e colunas de uma tabela da interface com as conexões da classe representada pela instância.

- Método `getConnectionsList`: chamado pelo método listConnections. Retorna a lista de conexões associadas ao módulo que representa a classe cujo nome é passado como parâmetro.

- Método `initComponents`: chamado pela construtora. Inicializa e configura todos os elementos da interface gráfica.

- Método `jButton1ActionPerformed`: determina a ação realizada por um botão da interface gráfica. Apenas evoca uma outra classe de interface que irá detalhar a conexão escolhida na tabela de conexões.

- Método `getClassModuleByName`: chamado pela ação do botão 1. Retorna o módulo, instância de ClassModule, cujo nome foi recebido como parâmetro.

- Método `jButton3ActionPerformed`: determina a ação realizada por um botão da interface gráfica. Apenas encerra esta classe e seu frame de interface, reativando o frame anterior que evocou a abertura desta instância.

Avaliação Geral de Coesão

Levando em consideração que praticamente todos os métodos relacionam-se de forma a

alcançar o objetivo único de extrair informações das classes envolvidas e exibi-las na interface, o grau de coesão da classe pode ser considerado alto. As exceções são os métodos que estabelecem as ações dos botões, cuja função é conectar este frame à cadeia de frames da interface. São, porém, absolutamente necessários. Sem eles, todas essas informações teriam de ser exibidas junto com outros conjuntos de informações não relacionadas em outro frame, como parte dele, o que prejudicaria seriamente a coesão do sistema como um todo.

Avaliação segundo LCOM

Valor 0. Máximo valor de coesão. Significa que existem, na classe avaliada, mais pares de métodos que se relacionam do que pares de métodos que não se relacionam por meio do uso em comum de atributos da classe.

Avaliação segundo a Coesão de Interesse

Valor 1. Máximo valor de coesão. Significa que, de acordo com a definição de interesse estabelecida pela métrica, apenas um interesse foi encontrado na classe avaliada.

Análise Comparativa

Ambos os valores foram satisfatoriamente coerentes com a Avaliação Geral. Como indicaram, tanto LCOM quanto Coesão de Interesse, o valor máximo de coesão conforme suas escalas, o teste foi útil para demonstrar um bom julgamento por parte das métricas, mas pouco contribui para sua comparação. Os botões, ao contrário do que se detectou na observação geral, não prejudicaram a coesão sob o ponto de vista das avaliações das métricas.

3.4.5 frmSelectFiles.java

- Construtora `frmSelectFiles`: apenas inicializa os elementos da interface por meio de outro método, initComponents, e define um dos campos da classe, o que se refere ao frame anterior.

- Método `initComponents`: chamado pela construtora. Inicializa e configura todos os elementos da interface gráfica.

- Método `jButton1ActionPerformed`: realiza a análise de um conjunto de classes que já deve ter sido selecionado, criando o grafo do sistema, computando dados, coletando valores de métricas e exibindo o resultado, tudo realizado por meio de funções posteriormente declaradas e definidas na classe ou declaradas e definidas em outras classes.

- Método `computeCoupling`: chamado pela ação do botão 1. Computa acoplamentos para todos os pares de classes do conjunto selecionado.
- Método `computeExpectedChangesByModules`: declara algumas variáveis mas não as utiliza. Não executa ação alguma.
- Método `jButton4ActionPerformed`: remove um subconjunto de classes do conjunto selecionado para análise.
- Método `jButton2ActionPerformed`: encerra este frame, esta instância da classe, e reativa o frame anterior.
- Método `jButton3ActionPerformed`: possibilita a seleção de um conjunto de arquivos .class, bytecodes de java, a serem analizados.
- Método `extractDirectoryFiles`: chamado pela ação do botão 3. Extrai recursivamente os arquivos .class que estejam dentro de um diretório.
- Método `showResultSystem`: chamado pela ação do botão 1. Evoca a criação do frame seguinte, responsável por exibir as informações do sistema (conjunto de classes avaliado), e encerra este.

Avaliação Geral de Coesão

Pode-se observar uma variedade de funções sendo realizadas pelos métodos, de forma que existe sim um relacionamento entre muitos deles mas não há um foco em determinada tarefa. Há métodos que dedicam-se a manipulações da interface e simultaneamente ao cálculo de métricas e construção de modelos do sistema. Essa gama de atividades sendo executadas em uma mesma classe prejudica a coesão do módulo e diminui a compreensibilidade do código. Além disso, há um método vazio na classe que não se relaciona com os outros de forma alguma, o que também prejudica a coesão da classe.

Avaliação segundo LCOM

Valor 13. Considerando que temos 10 métodos na classe, o valor máximo que LCOM poderia obter, indicando o pior nível possível de coesão, é a combinação de dez, dois a dois, que é dada por 10! / (2! * (10-2)!) que é igual a 45. Ou seja, LCOM forneceu 13 em 45, o que equivale a 29% de "não-coesão".

Avaliação segundo a Coesão de Interesse

Valor 0,333. Significa que, de acordo com o conceito de interesse estabelecido pela métrica, três interesses foram encontrados na classe avaliada.

Análise Comparativa

Ambas as métricas foram eficazes ao perceber uma queda na coesão devido à gama de atividades sendo realizadas pela classe e, provavelmente, também devido ao método vazio. Não há como afirmar qual delas foi mais próxima do que se esperava com a avaliação geral de coesão, já que as escalas são diferentes e a avaliação geral não pode ser realizada quantitativamente.

3.5 Classes Desenvolvidas por Alunos Iniciantes em OO

3.5.1 Professores.java

- Construtoras `Professores`: há duas funções construtoras. Uma delas possui um parâmetro a menos e inicializa um dos campos da classe com um valor padrão. A outra recebe o valor desse campo em um parâmetro. Ambas inicializam todos os campos da classe.
- Método `getCodigo`: apenas retorna o valor do campo codigo da classe.
- Método `setNome`: atribui ao campo nome o valor recebido como parâmetro.
- Método `getNome`: apenas retorna o valor do campo nome da classe.
- Método `setCpf`: atribui ao campo cpf o valor recebido como parâmetro.
- Método `getCpf`: apenas retorna o valor do campo cpf da classe.
- Método `setTitulacao`: atribui ao campo titulacao o valor recebido como parâmetro.
- Método `getTitulacao`: apenas retorna o valor do campo titulcao da classe.
- Método `setSituacao`: atribui ao campo situacao o valor recebido como parâmetro.
- Método `getSituacao`: apenas retorna o valor do campo situacao da classe.

Avaliação Geral de Coesão

Todos os métodos estão fortemente relacionados pois possuem o objetivo único de manipulação de dados pertinentes ao objeto representado pela classe, o professor, seja estabelecendo valores para esses dados ou recuperando os valores por eles assumidos. Além disso, apesar de não terem sido expostos aqui, todos os atributos da classe são privados, o que possibilita atribuir a ela um grau máximo de coesão.

Avaliação segundo LCOM

Valor 19. Considerando que temos 11 métodos na classe, o valor máximo que LCOM poderia obter, indicando o pior nível possível de coesão, é a combinação de onze, dois a dois, que é dada por 11! / (2! * (11-2)!) que é igual a 55. Ou seja, LCOM forneceu 19 em 55, o que equivale a 35% de "não-coesão".

Avaliação segundo a Coesão de Interesse

Valor 1. Máximo valor de coesão. Significa que, de acordo com a definição de interesse estabelecida pela métrica, apenas um interesse foi encontrado na classe avaliada.

Análise Comparativa

A métrica de Coesão de Interesse forneceu uma avaliação muito mais consistente com o esperado. O valor obtido corresponde ao máximo valor em sua escala, exatamente como constatou-se na avaliação geral. LCOM, por outro lado, mostrou-se bastante ineficaz na análise do módulo, indicando uma elevada ausência de coesão e, portanto, incoerente com o que se pode observar na classe em termos de coesão interna.

3.5.2 Main.java

- Método `Main`: define uma série de vetores para armazenar as entidades que irão representar uma universidade na visão do programa. Utiliza todo o conjunto de métodos definidos posteriormente na classe com o intuito de possibilitar a administração dessas entidades.
- Método `menuPrincipal`: exibe o menu principal do sistema e capta sua entrada.
- Método `menuCadastro`: exibe o menu de cadastros e capta sua entrada.
- Método `menuExibicao`: apresenta o menu de exibição e capta sua entrada.
- Método `menuCadTurma`: exibe o menu de cadastro de turmas e capta sua entrada.
- Método `menuAlteracao`: exibe o menu de alterações e capta sua entrada.
- Método `menuPesquisa`: exibe o menu de pesquisas e capta sua entrada.
- Método `cadastrarAluno`: cria um novo aluno no vetor de alunos.
- Método `cadastrarProfessor`: cria um novo professor no vetor de professores.
- Método `cadastrarDisciplina`: cria uma nova disciplina no vetor de disciplinas.
- Método `cadastrarTurma`: cria uma nova turma no vetor de turmas.

- Método `matricularAluno`: associa um aluno a uma turma.
- Método `cancelarMatricula`: desassocia um aluno a uma turma.
- Método `alterarAluno`: usado para modificar nome ou situação de alunos.
- Método `alterarProfessor`: usado para modificar vários dados de professores.
- Método `alterarDisciplina`: usado para modificar descrição ou situação de disciplinas.
- Método `alterarTurma`: usado para modificar vários dados de turmas.
- Método `pesquisarAlunos`: usa outros métodos definidos posteriormente para pesquisar alunos seguindo diferentes critérios.
- Método `procurarAlunosPorNome`: pesquisa alunos por nome e exibe o resultado por meio de método posteriormente definido.
- Método `procurarAlunosPorMatricula`: pesquisa alunos por matrícula e exibe o resultado por meio de método posteriormente definido.
- Método `procurarAlunosPorSituacao`: pesquisa alunos por situação e exibe o resultado por meio de método posteriormente definido.
- Método `pesquisarProfessores`: usa outros métodos definidos posteriormente para pesquisar professores seguindo diferentes critérios.
- Método `procurarProfessoresPorNome`: pesquisa professores por nome e exibe o resultado por meio de método posteriormente definido.
- Método `procurarProfessoresPorCodigo`: pesquisa professores por código e exibe o resultado por meio de método posteriormente definido.
- Método `procurarProfessoresPorCpf`: pesquisa professores por CPF e exibe o resultado por meio de método posteriormente definido.
- Método `procurarProfessoresPorTitulacao`: pesquisa professores por titulação e exibe o resultado por meio de método posteriormente definido.
- Método `procurarProfessoresPorSituacao`: pesquisa professores por situação e exibe o resultado por meio de método posteriormente definido.
- Método `pesquisarDisciplinas`: usa outros métodos definidos posteriormente para pesquisar disciplinas seguindo diferentes critérios.

- Método `procurarDisciplinasPorDescricao`: pesquisa disciplinas por descrição e exibe o resultado por meio de método posteriormente definido.
- Método `procurarDisciplinasPorCodigo`: pesquisa disciplinas por código e exibe o resultado por meio de método posteriormente definido.
- Método `procurarDisciplinasPorSituacao`: pesquisa disciplinas por situação e exibe o resultado por meio de método posteriormente definido.
- Método `pesquisarTurmas`: usa outros métodos definidos posteriormente para pesquisar turmas seguindo diferentes critérios.
- Método `procurarTurmasPorCodigo`: pesquisa turmas por código e exibe o resultado por meio de método posteriormente definido.
- Método `procurarTurmasPorAno`: pesquisa turmas por ano e exibe o resultado por meio de método posteriormente definido.
- Método `procurarTurmasPorSemestre`: pesquisa turmas por semestre e exibe o resultado por meio de método posteriormente definido.
- Método `procurarTurmasPorDisciplina`: pesquisa turmas por disciplina e exibe o resultado por meio de método posteriormente definido.
- Método `procurarTurmasPorProfessor`: pesquisa turmas por professor e exibe o resultado por meio de método posteriormente definido.
- Método `exibirAlunos`: imprime no terminal dados dos alunos.
- Método `exibirAlunosDaTurma`: faz exatamente a mesma coisa que o método exibirAlunos. Provável erro.
- Método `exibirProfessores`: imprime no terminal dados dos professores.
- Método `exibirDisciplinas`: imprime no terminal dados das disciplinas.
- Método `exibirTurmas`: imprime no terminal dados das turmas.
- Método `localizarAluno`: encontra a posição no vetor de alunos a partir da matrícula.
- Método `localizarProfessor`: encontra a posição no vetor de professores a partir do código.

- Método `localizarDisciplina`: encontra a posição no vetor de disciplinas a partir do código.
- Método `localizarTurma`: encontra a posição no vetor de turmas a partir do código.
- Método `localizarAlunoNaTurma`: encontra a posição de um aluno no vetor de alunos de uma turma.
- Método `cls`: imprime 15 linhas vazias no terminal.
- Método `mensagem`: imprime uma string e aguarda um intervalo de tempo.

Avaliação Geral de Coesão

Devido à extensão da classe, há uma dificuldade maior em estabelecer com precisão os fatores que unem os métodos. Com exceção dos métodos cls e mensagem, todos os outros realizam funções interessantes ao funcionamento do sistema simplificado de administração de uma universidade. Por não se desviarem desse objetivo, suas funções estariam relacionadas e, portanto, coesas. Porém, há métodos que conjugam apresentações para interface e manipulações de dados, o que contribui para a redução da coesão. Os métodos cls e mensagem não seguem a premissa de todos os anteriores, sendo responsáveis por detalhes cosméticos à apresentação. Isso significa que seu objetivo básico diverge do objetivo básico dos outros e, portanto, prejudicam a coesão da classe.

Avaliação segundo LCOM

Valor 675. Considerando que temos 49 métodos na classe, o valor máximo que LCOM poderia obter, indicando o pior nível possível de coesão, é a combinação de quarenta e nove, dois a dois, que é dada por 49! / (2! * (49-2)!) que é igual a 1176. Ou seja, LCOM forneceu 675 em 1176, o que equivale a 57% de "não-coesão".

Avaliação segundo a Coesão de Interesse

Valor 0,1. Significa que, de acordo com o conceito de interesse estabelecido pela métrica, dez interesses foram encontrados na classe avaliada.

Análise comparativa

Ambas as métricas foram capazes de perceber falhas de coesão no módulo. Eram esperados, de acordo com a avaliação geral, resultados com um nível mediano de coesão, na medida em que não puderam ser determinados motivos suficientemente concretos e fortes para abatê-la severamente. Porém, obtivemos para essa classe os menores valores de coesão encontrados até agora. Como já explicado, o tamanho desse módulo prejudica a realização de uma avaliação não automatizada e, portanto, é perfeitamente aceitável

essa discordância conjunta das métricas. Não é possível, pelo mesmo motivo, afirmar qual das métricas foi mais precisa em sua avaliação, o que torna o experimento pobre em possibilidades de comparação das métricas.

3.6 Resultados

A Tabela 3.1 resume todas as análises realizadas nos experimentos. Uma métrica é considerada *Correta* quando a idéia transmitida pelo valor que fornece é coerente com a idéia transmitida pela avaliação geral. Ela é considerada *Precisa* quando o valor que obtem é coerente com algum valor estabelecido na avaliação geral. Quando não se pode extrair da avaliação geral o valor para algum aspecto observado pelas métricas, então nada se pode afirmar sobre a precisão delas. Para efeito de organização, Coesão de Interesse é referenciada, nesta tabela, como CoIn.

Nome Classe	LCOM Correta	LCOM Precisa	CoIn Correta	CoIn Precisa
RealizaConsulta.java	Sim	Sim	Sim	Não
ClassModule.java	Não	Não	Sim	Sim
ConnectionPath.java	Sim	Não	Sim	Sim
frmClassDetail.java	Sim	Sim	Sim	Sim
frmSelectFiles.java	Sim	???	Sim	???
Professores.java	Não	Não	Sim	Sim
Main.java	Sim	???	Sim	???

Tabela 3.1: Resumo das Análises

Os experimentos revelaram, a grosso modo, que a precisão e a correção dos valores obtidos para a métrica de Coesão de Interesse são levemente superiores à precisão e à correção dos valores obtidos para a métrica LCOM. Porém, deve-se ressaltar que a base de testes não foi muito extensa e que, portanto, não se deve descartar a possibilidade de se constatar um comportamento diferenciado das métricas quando aplicadas sobre um conjunto muito maior de classes.

3.7 Avaliação Crítica

Para que se possa consolidar o conhecimento necessário para se utilizar adequadamente o sistema de avaliação de cada métrica, são, a seguir, destacados os seus pontos fracos. Ao exibir as falhas de cada uma delas, o objetivo é determinar o que as leva a exibir resultados incoerentes com uma avaliação de coesão independente e, dessa forma, determinar também em

que cenários o seu uso torna-se inadequado.

3.7.1 Falhas da Métrica LCOM

Desagregação de Atributos

Por não possuir uma propriedade de transitividade, LCOM pode ocasionar um efeito ilusório de desagregação de atributos da classe. Como o principal parâmetro da métrica é dado pelo número de pares de métodos relacionados ou não por meio do uso de atributos da classe em comum, classes coesas com métodos que utilizam atributos diferentes podem ser reconhecidas como classes não coesas. Isso não seria uma falha se realmente não fosse possível criar classes coesas com métodos que utilizam, em grande parte, atributos distintos. Acontece que isso não só é possível como é bastante comum. Imagine um Tipo Abstrato de Dados(TAD) qualquer que armazena informações sobre um determinado objeto ou entidade do mundo real, por exemplo, sobre uma pessoa. Poderia ser interessante, para o contexto do problema, registrar nesse TAD informações distintas como sexo(gênero), idade, peso e altura e, consequentemente, métodos para obtenção independente de cada uma delas. Cada um desses métodos iria lidar com um atributo diferente da classe, mas isso, de forma alguma, tornaria a classe pouco coesa. Ainda assim poderíamos avaliar a classe e constatar que sua coesão é absolutamente perfeita. LCOM, no entanto, já indicaria um baixo nível de coesão.

Falsos Negativos

A subtração realizada pela métrica ocorre apenas no caso de haverem mais métodos não relacionados do que métodos relacionados. Caso isso não ocorra, o valor obtido por meio da métrica é zero, ou seja, a ausência de coesão é nula. Esse sistema peca na medida em que simplesmente ignora um número pequeno de métodos não relacionados que venham a penalizar a coesão da classe. Imagine uma classe que possui cinco(5) métodos, dos quais quatro(4) são totalmente relacionados enquanto um(1) é totalmente não relacionado, e, além disso, prejudica seriamente a coesão da classe. Temos, nesse caso, seis(6) pares de métodos que se relacionam e quatro(4) pares de métodos que não se relacionam. Apesar de, como determinado, a coesão da classe ser prejudicada pelo método não relacionado, a avaliação segundo a métrica LCOM indicaria perfeita coesão na classe já que quatro(4) é menor que seis(6) e o valor obtido seria zero(0). Isso constituiria falso negativo pois seria indicação de nenhuma ausência de coesão, quando na verdade há ausência de coesão.

3.7.2 Falhas da Métrica Coesão de Interesse

Falsos Positivos/Relacionamento Incompleto

O conceito de relacionamento utilizado pela métrica encontra-se, ainda, incompleto. Ela toma como relacionados métodos de uma dada classe que utilizam os mesmos atributos dessa classe ou fazem chamadas aos mesmos métodos dessa classe. Há ainda casos em que um método chama outro método da classe e é o único método a evocá-lo, ocorrendo então uma composição de funcionalidades que pode apenas estar contribuindo para a execução de uma funcionalidade maior. Isso deveria incluir os dois métodos em um mesmo interesse. A métrica de Coesão de Interesse, porém, não detecta esse tipo de relacionamento e em consequência disso pode vir a criar mais interesses do que deveria. Dessa forma, pode haver uma indicação de falha de coesão quando na verdade não há falha alguma, ou seja, podem ocorrer falsos positivos.

3.8 Aperfeiçoamento da Métrica de Coesão de Interesse

A confiabilidade de uma métrica depende diretamente da precisão de seus critérios de avaliação. Além disso, como as métricas são frutos do conhecimento científico empírico, estão sujeitas a constante evolução. A métrica proposta por Ferreira et al. (8) e recém implementada pelo autor na ferramenta CONNECTA não escapa às mesmas regras. Por ter sido concebida há um curto período de tempo, são grandes as chances de que experimentos venham a revelar possíveis falhas e, por conseguinte, possíveis ajustes devam ser realizados sobre seus critérios de avaliação.

3.8.1 Motivação

Conforme explicitado nas Seções 3.6 e 3.7.2, uma aparente falha foi detectada na avaliação realizada com a métrica de Coesão de Interesse, o que a levou a exibir resultados imprecisos em alguns testes. Em algumas situações específicas a métrica indica um número maior de interesses do que se pode observar, de fato, nas classes. Isso faz com que o valor final fornecido indique uma coesão inferior à coesão real da classe. Como constatado, a falha decorre de uma imprecisão em um dos conceitos utilizados pela métrica, o conceito de relacionamento.

Relacionamento: dois métodos de uma classe C estão relacionados se utilizam pelo menos um atributo da classe C em comum ou pelo menos um método da classe C em comum (8).

Os relacionamentos são submetidos à transitividade e os conjuntos disjuntos de métodos relacionados compõem os interesses da classe. O caso identificado pelos experimentos, porém, exibia uma configuração que deveria compor um único interesse se considerássemos o conceito original de interesse, *um conjunto de funções relacionadas para a implementação de um propósito comum*, que é, inclusive, o que o conceito de interesse da métrica visa formalizar. Uma classe possuia vários métodos relacionados pelo uso de variáveis e métodos da própria classe em comum, mas um método em particular apenas evocava um dos outros e era o único método a fazê-lo. Sua funcionalidade era composta pela própria funcionalidade do método que chamava, para que um objetivo maior fosse alcançado, mas ainda sim não havia um outro método evocando aquele mesmo método. Por esse motivo, foi isolado em um interesse a parte, incapaz de se relacionar com os demais métodos, devido ao conceito de relacionamento adotado. Por fim, com um número de interesses formados superior ao real, a métrica exibiu um resultado que indicava falha de coesão onde não havia falha alguma.

3.8.2 Ajuste

Fica claro que é necessário estender o conceito de relacionamento para o seguinte:

Relacionamento: dois métodos de uma classe C estão relacionados se utilizam pelo menos um atributo da classe C em comum ou pelo menos um método da classe C em comum ou se um deles utiliza o outro.

A crença na simplicidade dos ajustes demandados, em vista da simplicidade de colocá-los em termos verbais, é um terrível engano. CONNECTA, como já exposto neste documento, realiza suas medições com base na análise do *bytecode* de classes Java. *Bytecodes*, porém, são surpreendentemente traiçoeiros. A Constant Pool, por exemplo, resiste bravamente à revelação de seus segredos, revelações essas necessárias para que sejam implementadas funções que interpretem adequadamente as referências encontradas ao longo do código. Quando chamadas de métodos ocorrem em meio ao *bytecode* de um método, por exemplo, há referências a posições da Constant Pool que contêm referências para outras posições que por fim contêm referências à posição onde encontramos, após muita persistência, o nome do método evocado. O interessante é tentar induzir isso quando não há uma documentação de *bytecode* Java disponível. Difícil afirmar qual é mais intensa, se a frustração com as insistentes referências inconsistentes ou se a realização alcançada com a descoberta do tortuoso caminho que as torna consistentes.

3.9 Conclusões

A busca pela manutenibilidade de um software deve passar pela necessidade de se alcançar um alto grau de coesão. Quanto mais coesos são os seus módulos, menor é a conectividade do programa pois menor é acoplamento entre suas partes. Além de maximizar o reúso e proporcionar outros benefícios, alta coesão minimiza os esforços de modificação, o que facilita a manutenção, atividade responsável pela maior parte dos custos totais de um sistema. Para se alcançar um nível elevado de coesão é, então, necessária a instrumentação adequada para que se possa avaliá-la.

Por meio de experimentos foram comparadas avaliações de classes sob a ótica da métrica mais popularmente utilizada, LCOM, e sob a ótica da Coesão de Interesse. Os resultados evidenciaram falhas em ambos os sistemas de medição mas comprovaram uma leve superioridade da nova métrica. Por fim, para a falha encontrada em um dos conceitos da métrica de Coesão de Interesse, uma solução foi proposta e implementada, aumentando a confiabilidade dos resultados exibidos pelo programa.

4 Reestruturação de CONNECTA

Esta seção apresenta um conjunto de facilidades que foram incorporadas em CONNECTA, pelo autor, após a implementação da métrica de coesão de interesse e após a realização dos testes de análise comparativa.

4.1 Implementação de Uma Nova Forma de Execução para CONNECTA

CONNECTA é um software desenvolvido na linguagem de programação Java. Como tal, pode ser executado pela linha de comando evocando-se a ferramenta "java", que é um lançador de aplicações escritas em Java. Porém, o comando a ser executado, no caso particular do CONNECTA, é excessivamente complexo e envolve opções cujo alto grau de especificidade as torna desconhecidas para o público alvo do programa. Isso cria a necessidade de se estabelecer uma forma alternativa de executar o programa, algo que seja mais simples e mais claro para o usuário.

Programas Java precisam definir explicitamente, na própria linha de comando para sua execução, as classes java que utilizarão, para que estas sejam carregadas inicialmente pela máquina virtual. CONNECTA baseia todas as suas funções na leitura de classes Java e, portanto, o usuário precisa definir na linha de comando qual o diretório raíz do programa que será lido. Caso isso não seja feito, CONNECTA ainda sim será executado e exibirá suas funcionalidades aparentemente completas. Porém, as análises realizadas não irão obter sucesso, pois as classes não serão encontradas, já que não foram carregadas pela máquina virtual. Isso gera um outro problema: a execução de CONNECTA precisa ser encerrada e reiniciada, com um novo caminho para classes, a cada vez que se queira avaliar um outro programa.

4.1.1 Solução Adotada

A solução encontrada para ambos os problemas descritos, após uma cuidadosa análise das possibilidades de ação, foi o desenvolvimento de um sistema cuja execução engloba a execução do CONNECTA. Devido ao fato de evocar instâncias de CONNECTA, ele foi denominado CONNECTA CALLER.

CONNECTA CALLER é um software simples, que trabalha em conjunto com o CONNECTA original, e cuja função é construir a linha de comando, a partir de escolhas realizadas pelo usuário em sua interface gráfica, e executá-la em um processo distinto interno. Isso torna a execução do CONNECTA muito mais simples do ponto de vista do usuário, bastando indicar o diretório onde se encontra o seu .jar e o diretório raíz do programa a ser analisado.

4.1.2 Implementação

As seguintes classes Java formam a base para a criação de subprocessos, permitindo a execução de uma linha de comando em algum ponto da execução de um programa.

java.lang.Runtime: a função Runtime.getRuntime() é chamada para que se obtenha a instância de objeto runtime associada à aplicação corrente. A instância de runtime é que permite à aplicação interfacear com o ambiente em que é executada. De posse desse objeto, é possível evocar sua função exec(String comando), que executa o comando especificado em um processo separado e retorna esse processo.

java.lang.Process: o processo criado para a execução da linha de comando precisa ter suas saídas redirecionadas ou não será executado corretamente. Isso é feito utilizando-se os métodos de instância de Process, getInputStream() e getErrorStream().

java.io.BufferedReader: usado para captar as saídas do processo criado na execução da linha de comando.

java.io.InputStream: também usado para captar as saídas do processo criado na execução da linha de comando.

java.io.InputStreamReader: também usado na captação das saídas do processo criado na execução da linha de comando.

4.2 Implementação de Suporte a Banco de Dados

A atividade de avaliação, independentemente do tipo de alvo, tem como objetivo primordial o conhecimento do atual estado para que esse possa vir a ser aprimorado. Avaliar é, inerentemente, identificar os caminhos para possíveis melhorias. Porém, não se toma conhecimento da melhora obtida seguindo-se determinado caminho se resultados de avaliações anteriores não são registrados e mantidos. Portanto, é de vital importância para qualquer sistema de avaliação o suporte para registro de informações e resultados obtidos.

CONNECTA é, essencialmente, uma ferramenta de avaliação, que implementa diversos padrões de medição. Para que as medições realizadas realmente se concretizem em melhorias sobre os programas avaliados, é necessário que a ferramenta seja capaz de registrar seus resultados para posterior consulta e comparação. Isso já havia sido implementado, usando-se um arquivo para a escrita do conjunto de resultados da avaliação de um sistema. Esse método, porém, não garante a segurança da informação, já que qualquer usuário poderia vir a manipular os arquivos de registro de avaliação gerados. Além disso, a informação não estaria suficientemente organizada e condensada, de modo a permitir o mínimo gasto de memória. Baseando-nos nesses fatos e considerando a grande quantidade de informações geradas pela ferramenta, optamos pela implementação de suporte a um sistema de gerenciamento de bancos de dados. O sistema escolhido foi o MySQL.

4.2.1 Gerenciador de Banco de Dados MySQL

MySQL é um sistema de gerenciamento de bancos de dados(SGBD) que utiliza a linguagem SQL como interface. É um software livre, desenvolvido em código aberto, reconhecido por seu desempenho e robustez e também por ser multi-tarefa e multi-usuário. Outro grande motivo para sua escolha como sistema utilizado pelo CONNECTA foi o fato de o autor já possuir alguma experiência prévia em sua utilização.

4.2.2 Implementação

A implementação de suporte a um SGBD pode ser dividida em três grandes etapas:

Desenvolvimento do Esquema Relacional: compreende o planejamento de todo o banco, e seu produto é um esquema relacional que informa quais serão suas tabelas, quais serão os atributos de cada uma delas e por meio de quais atributos as tabelas irão se relacionar. Este estágio não coube ao autor, sendo a aluna de doutorado Kecia Marques responsável

por sua realização. O resultado dessa etapa é o esquema da Figura 4.1.

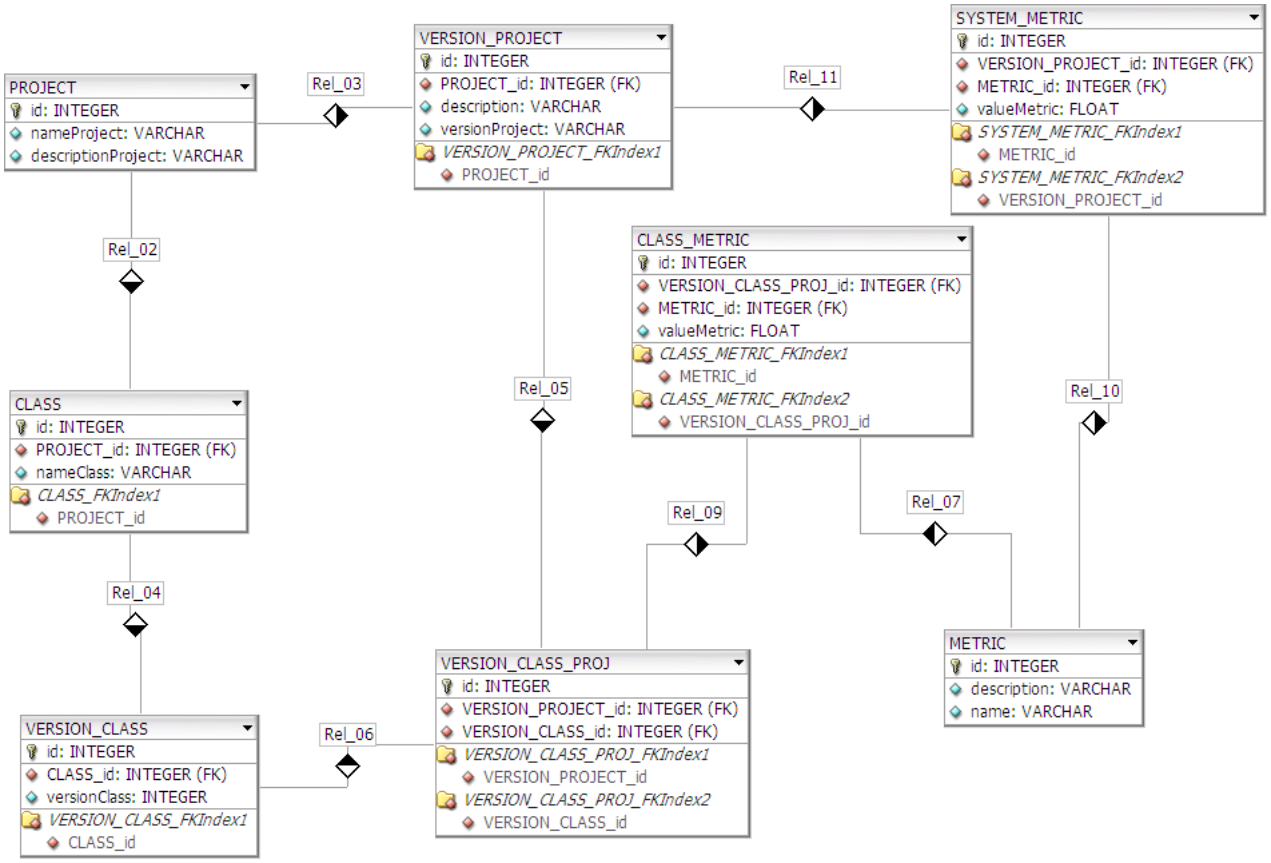

Figura 4.1: Esquema Relacional para o Banco de Dados de CONNECTA

Desenvolvimento dos Scripts: compreende a confecção de arquivos que agrupam comandos a serem executados pelo SGBD para criar usuários para os bancos de dados, conceder-lhes privilégios de acesso e criar os bancos de dados propriamente ditos, e para remover todos os registros criados.

Desenvolvimento do Código: compreende a confecção de classes ou funções da aplicação para acesso aos bancos de dados criados.

4.2.3 Desenvolvimento dos Scripts

Toda operação em um sistema de gerenciamento de bancos de dados é realizada por um usuário. A ferramenta CONNECTA, procurando evitar a necessidade de fornecimento de informações confidenciais por parte do usuário, utiliza nome e senha de usuário próprios. Esse usuário é criado por meio do script de criação, que deve ser executado pelo administrador do servidor do SGBD ou por um usuário que possua as mesmas permissões de criação do administrador. Após criar o usuário, o script cria o banco de dados a ser utilizado pelo CONNECTA

e cria nesse banco as tabelas para registros de projetos, versões de projetos, classes, versões de classes, associações de versões de classes a versões de projetos, métricas gerais, métricas de sistema e métricas de classes.

O conjunto de métricas a serem registradas no banco precisa ser determinado já no script de criação, pois a etapa de codificação vai depender de algumas constantes determinadas na inclusão desses registros. Por esse motivo, o script é também responsável pelo povoamento da tabela de métricas.

Um segundo script é responsável pela remoção de todo e qualquer registro realizado no banco de dados pelo script de criação e pelas atividades do CONNECTA, removendo o usuário criado e o banco de dados, com todas as suas tabelas e os registros constantes em cada uma delas.

O script de criação deve ser executado como parte da instalação de CONNECTA caso o usuário tenha a intenção de utilizar os seus serviços de bancos de dados e o script de destruição deve ser executado como parte da desinstalação da ferramenta.

4.2.4 Desenvolvimento do Código

Uma série de classes foram criadas e adicionadas ao CONNECTA para conferir-lhe a capacidade de interação com o MySQL. As funções utilizadas em sua codificação pertencem à interface de programação de aplicativos(API) JDBC.

MySQL Connector & JDBC

JDBC é um conjunto de classes e interfaces escritas em Java que faz o envio de instruções SQL para qualquer banco de dados relacional, possibilitando o uso em aplicações Java de bancos de dados já instalados. Como diferentes tipos de bancos de dados não necessariamente utilizam os mesmos protocolos de acesso, leitura, escrita etc, é necessário um driver específico que traduza as chamadas JDBC para o protocolo utilizado pelo sistema de gerenciamento de bancos. No caso do MySQL, o driver a ser utilizado é o MySQL Connector/J.

Novas Classes

RealizaConsulta.java: cria uma conexão com o banco de dados cujo nome é passado como parâmetro, usando o nome de usuário e a senha passados como parâmetros. Possui métodos para realizar consultas e para realizar atualizações no banco de dados. É a principal

base para a construção das funcionalidades de todas as outras classes relacionadas ao suporte a bancos de dados.

frmConsultaBancodeDados.java: classe de interface gráfica para consulta e exibição dos projetos e suas respectivas versões.

frmDetalhesProjetoBD.java: classe de interface gráfica para exibição de valores de métricas de um projeto e exibição de ids, nomes e versões de classes a ele relacionadas. Consulta de detalhes de classe por nome.

frmDetalhesClasseBD.java: classe de interface gráfica para exibição de versões e valores de métricas de versões de uma classe.

Pegador.java: utiliza instâncias da classe RealizaConsulta para obter identificadores de registros específicos no banco de dados. Obtém o valor do atributo ID de: um projeto, versão de projeto, classe, versão de classe, associação de versão de projeto a versão de classe, métrica, métrica de versão de projeto ou métrica de versão de classe. Essencial para a composição das funcionalidades das classes dedicadas à escrita no banco de dados.

frmCadastrarEscolherProjeto.java: classe de interface gráfica para criação de um novo projeto ou escolha de um projeto existente.

Se um projeto e uma versão de projeto que não existem são escolhidos, são criados registros no banco de dados para: um projeto, uma versão de projeto, métricas dessa versão de projeto, classes desse projeto, versões de cada classe, associações de cada uma dessas versões de classe à nova versão do projeto e métricas de classe para essas associações.

Se um projeto que já existe e uma versão que não existe são escolhidos, são criados registros para: uma nova versão de projeto, as métricas dessa nova versão, as classes que ainda não constavam no projeto, as novas versões de todas as classes do projeto, as associações de cada uma dessas versões de classe à nova versão de projeto e métricas de classe para essas associações.

Se um projeto e uma versão de projeto que já existem são escolhidos, uma instância da classe frmEscolherClasses é criada e ativada.

frmEscolherClasses.java: classe de interface gráfica em que o usuário deve escolher quais das classes recém analisadas devem receber novas versões. As classes selecionadas que ainda não pertençam ao projeto são nele incluídas e uma nova versão de cada uma das classes selecionadas é criada e associada à versão do projeto. Métricas de classe são criadas para cada uma dessas associações.

4.3 Conclusão

Esta seção teve como foco as melhorias introduzidas na ferramenta CONNECTA. Uma nova forma de executá-la foi desenvolvida, tornando a tarefa mais simples, intuitiva e menos suscetível a falhas. O suporte a um sistema de gerenciamento de bancos de dados foi implementado, permitindo ao programa registrar de forma organizada grandes quantidades de resultados de avaliações e permitindo ao usuário acompanhar com mais precisão e agilidade o progresso das classes de seus sistemas.

5 CONCLUSÃO

Baseando-se nos fatos expostos neste relatório e nos resultados obtidos pelo autor ao longo do período de realização desta Iniciação Científica, destacamos as seguintes contribuições:

1. Implementação e aprimoramento de uma nova métrica para avaliação de coesão em módulos, denominada Coesão de Interesse.
2. Expansão de CONNECTA, com a integração da leitura da nova métrica à ferramenta, com a implementação do suporte a um sistema de gerenciamento de bancos de dados e com a criação de um caminho alternativo intuitivo de execução.
3. Correção de CONNECTA, após a detecção de falhas em seu código original.
4. Realização de um número expressivo de testes para coleta de valores de Coesão de Interesse, assim como valores de LCOM.
5. Avaliação de ambas as métricas LCOM e CoIn, comparando-as com uma avaliação qualitativa de coesão.

Com os resultados obtidos, surge a perspectiva para os seguintes trabalhos futuros:

1. Adição da capacidade de avaliação, em CONNECTA, de programas escritos em outras linguagens de programação.
2. Ampliação da base de testes utilizada na avaliação da correção e precisão das métricas Coesão de Interesse e LCOM.

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 STEVENS, W.; MYERS, G.; CONSTANTINE, L. Structured design. *IBM Systems Journal*, v. 13, p. 115–139, 1974.

3 CHIDAMBER, S. R.; KEMERER, C. F. Towards a metrics suite for object oriented design. In: *Proceedings of 6th ACM Conference on Object-Oriented Programming Systems Languages and Applications (OOPSLA)*. [S.l.: s.n.], 1991. p. 197–211.

4 SAN DIEGO STATE UNIVERSITY. *Advanced Object-Oriented Design and Programming*. [S.l.], mar. 2009. Disponível em: <http://www.eli.sdsu.edu/courses/spring98/cs635/notes/>.

5 MYERS, G. J. *Reliable Software through Composite Design*. New York: Petrocelli/Charter, 1975. 159 p. ISBN 0-88405-284-2.

6 UNIVERSITY OF OTTAWA. *Object Oriented Software Engineering*. [S.l.], fev. 2005. Disponível em: <http://www.site.uottawa.ca:4321/oose/index.html>.

7 MEYER, B. In: *Object-oriented software construction*. [S.l.: s.n.], 1997, (Prentice Hall International Series in Computer Science).

8 FERREIRA, K. A. M. et al. *Predição de Esforço de Manutenção de Software OO*. RT LLP001/2009, Departamento de Ciência da Computação/Universidade Federal de Minas Gerais, Belo Horizonte, fev. 2009.

9 FERREIRA, K. A. M. et al. Reference values for object-oriented software metrics. In: *Proc. IEEE XXIII Simpósio Brasileiro de Engenharia de Software (SBES'09)*. Fortaleza, Brasil: [s.n.], 2009. p. 62–72.

10 THE Sourceforge Website. 2009. Disponível em: <http://www.sourceforge.net>.