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

Heitor Corrêa de Almeida

MONOGRAFIA DE PROJETO ORIENTADO EM COMPUTAÇÃO I

Implementação e Análise Comparativa de Duas Métricas de Coesão em Software Orientado por Objetos

Belo Horizonte – MG
2009 / 1º semestre

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

Implementação e Análise Comparativa de Duas Métricas de Coesão em Software Orientado por Objetos

por

Heitor Corrêa de Almeida

Monografia de Projeto Orientado em Computação I

Apresentado como requisito da disciplina de Projeto Orientado em Computação I do Curso de Bacharelado em Ciência da Computação da UFMG

Profa. Dra. Mariza Andrade da Silva Bigonha
Orientadora

Ms. Kecia Aline Marques Ferreira
Co-orientadora

_____

Heitor Corrêa de Almeida - Aluno

_____

Mariza Andrade da Silva Bigonha - Orientadora

_____

Kecia Aline Marques Ferreira - Co-orientadora

Belo Horizonte – MG
2009 / 1º semestre

A Deus,
aos professores,
aos colegas de curso,
aos meus familiares,
ao Jackie Chan,
ao Yusuke Urameshi,
e a todas as crianças do mundo,
dedico este trabalho.

Agradecimentos

Inicialmente agradeço a Deus, pelas graças recebidas e pela chance na Terra.
Agradeço aos meus pais, pelo amor incondicional e dedicação à minha educação.
Aos meus professores, pelos conhecimentos adquiridos.
Aos colegas de curso pela convivência e trocas de experiências.
Ao Jackie Chan por mostrar como ser um cara legal independentemente das adversidades.
Ao Yusuke Urameshi por me ensinar a enfrentar com empolgação e determinação todos os obstáculos. Aliás, todos não. Só os interessantes.
E finalmente a todas as crianças do mundo por mostrarem o que nunca deve morrer em nós.

Sumário

Lista de Tabelas

Lista de Siglas

OO	*Orientação por Objetos, Orientado por Objetos*
LCOM	*Lack of Cohesion in Methods, Ausência de Coesão em Métodos*
CoCo	*Coesão Contratual*
TAD	*Tipo Abstrato de Dados*

Resumo

A avaliação de métricas é um importante recurso para o aprimoramento de sistemas de software. A capacidade de se comparar valores assumidos por determinadas características quantificáveis em programas diferentes permite que se destaquem pontos a serem alvo de possíveis melhorias.

Este trabalho implementa uma nova métrica para avaliar coesão em software orientado por objetos, denominada coesão contratual, e a compara com outra métrica disponível para esse 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.

Como parte desse trabalho, testes são realizados com o intuito de expressar qual das métricas retrata mais fielmente uma avaliação geral de coesão, além de pontos fracos e fortes de cada uma.

Palavras-chave: métricas, coesão, orientação por objetos, manutenibilidade, fidelidade de representação.

Abstract

The evaluation of metrics is an important resource for the improvement of software systems. The ability to compare the values taken on by certain quantitatively measurable features in different programs allows the highlighting of possible improvement points.

This paper implements a new metric to evaluate cohesion in object oriented software, called Contractual Cohesion, and compares it to another availeable metric for that purpose. These metrics evaluate how related are the internal elements of the modules that compose a program. The bigger the internal cohesion of modules, smaller are the coupling and the complexity of the system. Measuring cohesion is, therefore, a task of extreme relevance in the making of programs that may come to demand minimum maintenance costs. High maintainability is desirable since maintenance corresponds to the biggest slice of the system's total cost.

As part of this study, tests are executed with the intention of expressing which of the metrics portraits a general cohesion evaluation with the higher fidelity, besides strengths and weaknesses of each of them.

Keywords: metrics, cohesion, object-orientation, maintainability, representation fidelity.

1 INTRODUÇÃO

1.1 Visão Geral

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, 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.

Em vista da relevância da coesão na determinação dos custos de manutenção que, segundo Meyer (3), 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 (4), (5), (6), (7), a mais popular delas em relação à OO é a *Lack of Cohesion in Methods*(LCOM) (4), muito embora críticas plausíveis possam ser feitas à sua avaliação de coesão. Na Seção 4 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.

1.2 Objetivo, Justificativa e Motivação

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 Contratual. Essencialmente, essa métrica realiza a avaliação de um software com base no número de contratos estabelecidos por uma classe.

Baseado nos fatos expostos, nosso trabalho de pesquisa possui os seguintes objetivos:

- Implementação da métrica de Coesão Contratual.
- Incorporação dessa métrica à ferramenta CONNECTA (1).
- Realização de um número suficiente de testes para coleta de seus valores assim como valores de LCOM.
- Avaliação de ambas as métricas, comparando-as perante uma avaliação qualitativa de coesão.

Como resultado deste trabalho apresentamos as seguintes contribuições:

1. Implementamos a métrica Coesão Contratual e a incorporamos em CONNECTA.
2. Realizamos uma massa de testes, inicialmente usando o próprio CONNECTA, e, posteriormente outros softwares disponíveis.

Avaliamos as métricas LCOM e Coesão Contratual, comparando-as qualitativamente e, dessa forma, pudemos determinar a qualidade da informação fornecida por cada uma dessas métricas. Ressaltamos seus pontos fracos, ou seja, identificamos que situações podem provocar distorções nos resultados das análises realizadas de modo a torná-las incoerentes com o que se observa, de fato, do ponto de vista de um especialista, nas classes avaliadas.

2 REFERENCIAL TEÓRICO

O objetivo desta seção é fornecer alguns conceitos essenciais para o entendimento do processo de avaliação realizado pelas métricas e das comparações realizadas entre elas. Uma definição mais rica de coesão é necessária para consolidar a base do trabalho e permitir a construção dos conceitos e procedimentos das métricas.

2.1 Coesão

Coesão é a medida de quão fortemente relacionadas e concentradas estão as várias responsabilidades de um módulo de software. 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. Essa métrica de qualidade de software foi definida inicialmente por Myers et al. (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((4), (5), (6), (7)).

2.1.1 Ausência de Coesão em Métodos

LCOM(ausência de coesão em métodos) é uma das métricas do conjunto proposto por Chidamber e Kemerer (4) 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 formal 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

2.1.2 Contrato e Coesão Contratual

Para estabelecer a definição de Coesão Contratual (8), a métrica cuja implementação e avaliação constitui parte deste trabalho, é preciso primeiro definir o conceito de contrato. Um contrato é uma série de compromissos assumidos por classes que se relacionam, em um modelo de relacionamento *cliente-fornecedor*. A classe fornecedora compromete-se a fornecer um determinado serviço a partir de uma especificação do problema e publica a interface a ser utilizada por aquelas que precisarem utilizar o serviço. A classe cliente compromete-se a cumprir as pré-condições para utilizá-lo. Para benefício da modularidade, é importante a criação de classes que implementam um único contrato. Um exemplo clássico de programação por contrato é a implementação, em uma única classe, dos tipos abstratos de dados fila e pilha. Essa classe implementa dois contratos. 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 contrato.

A métrica coesão contratual baseia-se no número de contratos que uma classe implementa. Seu valor é então dado por $1/$*total de contratos da classe* . Se uma classe implementa dez contratos, por exemplo, o valor assumido pela métrica é 0.1. Da mesma forma, em uma classe que implementa apenas um contrato a métrica assume o valor 1. Ou seja, quanto mais próximo de 1 é o valor da métrica, melhor a sua coesão contratual. Quanto mais próximo de 0, pior. Para que possamos definir a métrica formalmente, precisamos então definir alguns conceitos (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**.

Contrato: os conjuntos disjuntos formados por todos os métodos relacionados entre si de uma classe compõem os contratos dessa classe. Ou seja, um contrato é um conjunto de métodos relacionados.

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

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

$Coesao = 0$, caso contrário

3 METODOLOGIA

O trabalho realizado baseia-se em um estudo teórico e experimental. O trabalho envolveu a implementação de algoritmo e realização 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 Procedimentos Metodológicos

Os valores para as métricas foram obtidos com a utilização da ferramenta CONNECTA, desenvolvida por Ferreira (1) em sua dissertação de mestrado. CONNECTA obtem os valores de diversas métricas relacionadas à avaliação de conectividade e manutenibilidade a partir do *bytecode* de programas escritos na linguagem Java. A implementação da leitura da métrica de coesão contratual foi realizada sobre esse software, tornando-o ainda mais completo e ampliando seu leque de possibilidades de uso.

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 (9), 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 considerada neste trabalho, o código fonte do próprio CONNECTA, já que o avaliador o teria conhecido a fundo durante a implementação da nova métrica.

O trabalho descrito neste texto foi realizado de acordo com as seguintes etapas:

Levantamento Bibliográfico: obtenção de um conhecimento mais profundo a respeito de coesão interna de módulos para possibilitar uma avaliação padrão que viria a servir de base para a crítica às métricas.

Implementação da Métrica de Coesão Contratual: implementação da métrica de coesão contratual na ferramenta CONNECTA.

Realização de Experimentos: avaliação geral de coesão e obtenção dos valores das métricas para um grupo de classes selecionadas.

Análise dos Resultados: compilação de todos os dados obtidos com os testes e formulação de conclusões quanto à informação fornecida pelas métricas LCOM e coesão contratual.

Elaboração do Texto de POC I: executada paralelamente a todas as outras.

4 TESTES E ANÁLISE DOS RESULTADOS

O texto a seguir descreve um conjunto de testes para comparação das avaliações de coesão realizadas segundo as métricas LCOM e Coesão Contratual. As métricas foram implementadas no sistema CONNECTA e as classes utilizadas pertencem ao próprio programa e foram selecionadas por constituírem exemplos de módulos não muito longos, adequados para uma avaliação humana, não automatizada, da coesão, cujo objetivo é servir de base para a crítica aos resultados das duas métricas. As duas últimas classes avaliadas pertencem a um outro sistema, desenvolvido por alunos iniciantes na programação OO, para adiministração simplificada de uma universidade.

4.1 Experimentos de Avaliação Comparativa entre as Métricas LCOM e Coesão Contratual

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 Contratual: valor de coesão obtido de acordo com a métrica Coesão Contratual.

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

Classes de CONNECTA

4.1.1 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 Contratual

Valor 0,5. Segundo melhor valor de coesão que pode ser obtido com a métrica. Ocorre porque o método Finalize foi considerado como um contrato 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 contratos, a métrica Coesão Contratual exibiu uma certa imprecisão em sua avaliação.

4.1.2 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 getCoesão Contratual: 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 Contratual

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 Contratual 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.

4.1.3 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 Contratual

Valor 0,5. Segundo maior valor de coesão. Significa que, de acordo com a definição de contratos estabelecida pela métrica, foram encontrados dois contratos 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 Contratual 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.

4.1.4 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 Contratual

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

Análise Comparativa

Ambos os valores foram satisfatoriamente coerentes com a Avaliação Geral. Como indicaram, tanto LCOM quanto Coesão Contratual, 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.

4.1.5 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 Contratual

Valor 0,333. Significa que, de acordo com o conceito de contrato estabelecido pela métrica, três contratos 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.

Classes desenvolvidas por alunos iniciantes em OO

4.1.6 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 Contratual

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

Análise Comparativa

A métrica de Coesão Contratual 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.

4.1.7 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 Contratual

Valor 0,1. Significa que, de acordo com o conceito de contrato estabelecido pela métrica, dez contratos 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.

4.2 Resultados

A Tabela 4.1 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 e 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 Contratual é referenciada, nesta tabela, como CoCo.

Nome Classe	LCOM Correta	LCOM Precisa	CoCo Correta	CoCo 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 4.1: Resumo das análises

Os experimentos revelaram, a grosso modo, que a precisão e a correção dos valores obtidos para a métrica de Coesão Contratual 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.

5 CONCLUSÕES

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.

5.1 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 o 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.

5.2 Falhas da Métrica Coesão Contratual

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 contrato. A métrica de Coesão Contratual, porém, não detecta esse tipo de relacionamento e em consequência disso pode vir a criar mais contratos 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.

5.3 Conclusões Gerais

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.

Neste trabalho, realizou-se com sucesso a implementação de uma nova métrica avali-

adora de coesão, Coesão Contratual, 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.

Por meio de experimentos foram comparadas avaliações de classes sob a ótica da métrica mais popularmente utilizada, LCOM, e sob a ótica da Coesão Contratual. Os resultados evidenciaram falhas em ambos os sistemas de medição mas comprovaram uma leve superioridade da nova métrica. Além disso, os testes apontaram uma possível melhoria a ser realizada, futuramente, sobre ela.

5.4 Trabalhos Futuros

A execução do presente trabalho abriu lacunas para os seguintes trabalhos futuros:

- **Aperfeiçoamento da Métrica de Coesão Contratual**
  Conforme discutido na descrição das falhas das métricas, o conceito de relacionamento utilizado na constituição dos contratos encontra-se incompleto. Seria interessante incluir o caso citado na avaliação da métrica, a verificação de chamadas únicas de métodos internos à classe para composição de funcionalidades e verificar se, de fato, a modificação aperfeiçoou seu poder de avaliação de coesão ou prejudicou-o.

- **Ampliação da Base de Testes**
  O número de experimentos realizados pode ser ampliado para tornar mais confiável a conclusão obtida a partir dos resultados.

- **Adição de Suporte a Outras Linguagens de Programação**
  O programa sobre o qual foram implementadas as verificações das métricas LCOM, Coesão Contratual dentre outras, CONNECTA, oferece suporte apenas a classes escritas na linguagem de programação Java. Seria útil estender suas funcionalidades para que passasse a oferecer suporte a outras linguagens.

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

4 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.

5 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/>.

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

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

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